UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE ESTUDIOS DE POSGRADO

"COMPENSACIÓN DE MOVIMIENTO POR REJILLAS DEFORMABLES USANDO ALGORITMOS DE APAREAMIENTO DE BLOQUES"

TESIS

PRESENTADA POR:

SALVADOR EUGENIO AYALA RAGGI

PARA OBTENER EL GRADO DE:

MAESTRO EN INGENIERÍA ELÉCTRICA (COMUNICACIONES)

Dirección: Dr. Francisco García Ugalde

CIUDAD UNIVERSITARIA

- Julio 1996 -

**TESIS CON
FALLA DE ORIGEN**

RESUMEN

La codificación de video digital es un área de investigación relativamente reciente, cuyos primeros estudios datan de la década de los sesentas. Esta tesis trata un tema muy especifico de la codificación de video, que es la compensación de movimiento, la cual consiste en "sintetizar" tramas de una secuencia de video a partir de sus predecesoras y de un conjunto de parámetros de movimiento estimados. La idea básica de este tipo de esquemas es aprovechar la redundancia existente entre tramas consecutivas, para transmitir mucho menos datos al receptor sin pérdida de calidad de la imágen. Existen diversos métodos de compensación de movimiento entre los cuales se encuentran los algoritmos de apareamiento de bloques, que son ampliamente utilizados en codificación de video en tiempo real, dada su gran sencillez operativa. Esta tesis revisa primero algunos de los ejemplos más populares de los métodos por apareamiento de bloques, y los utiliza como fundamento para desarrollar un sistema de compensación de movimiento basado en rejillas deformables, el cual emplea codiciones iniciales apropiadamente ajustadas, para obtener en poco tiempo de procesamiento una compensación libre de efectos de bloque, y con alta relación señal a ruido, además de una baja entropía para los parámetros de movimiento transmitidos.

Contenido

Capítulo 1

Introducción

La compensación de movimiento en los sistemas de codificación de video actuales, representa un aspecto de fundamental importancia para la obtención de altas tasas de compresión. Aprovechar de manera óptima la redundancia que existe entre los objetos de una trama y la siguiente en una secuencia de video, constituye un paso fundamental en una compresión de datos sin pérdida ni degradación de la calidad visual en una secuencia. Es así que la compensación de movimiento en base al buen conocimiento del mismo en la escena puede aprovechar las partes o pedazos de una trama anterior, ya sea trasladándolos, rotándolos o deformándolos, para formar con estos una "trama de predicción" lo más parecida posible a la trama siguiente. El objetivo de la compensación es entonces, construir esta trama de compensación con la mínima cantidad posible de parámetros de movimiento, ya que estos deben ser transmitidos al receptor para que éste pueda reconstruir dicha trama a partir de la trama anterior y los parámetros de movimiento. En la actualidad se estudian métodos matemáticos que modelan el movimiento tridimensional de los objetos para poder efectuar una mejor compensación.

En este trabajo de tesis se fijó como objetivo y se alcanzó obtener un esquema de compensación de movimiento que aprovecha los primeros métodos de compensación por apareamiento de bloques, de tal manera que se pueda alcanzar una notable mejora en los métodos de transformación geométrica por interpolación bilineal en rejillas de control.

El esquema propuesto utiliza algoritmos de apareamiento de bloques junto con una nueva técnica de correlación por vecindad llamada búsqueda sesgada para conseguir una estimación de movimiento óptima la cual se puede apreciar por la alta coherencia del campo de desplazamiento obtenido.

Posteriormente, se utiliza este campo de desplazamiento como condición inicial de los métodos de optimización por iterpolación en rejillas de control.

El fruto de este trabajo está principalmente en la mejora, en cuanto a la velocidad de cálculo, que los métodos de optimización por rejillas de control presentan, cuando comienzan a iterar con condiciones iniciales preestablecidas que en ocasiones pueden ser muy cercanas a las condiciones finales de tales procesos de optimización.

Además, también se obtiene una mejora sobre la entropía de los campos de desplazamiento de salida. Este es un aspecto muy importante, en virtud de que al fin y al cabo se está trabajando para alcanzar mejores niveles de compresión. El hecho de tener una baja entropía en el campo de vectores de movimiento (el cual como sabemos debe ser transmitido en este tipo de sistemas de compresion) contribuye aún más a la mejor codificación de la señal de video.

La calidad subjetiva de las imágenes compensadas con interpolación por rejilla de control, es además muy superior a la calidad de las imágenes compensadas por los métodos tradicionales de apareamiento de bloques, debido a la gran cantidad de artefactos de bloque que aquellos generan.

La interpolación por rejillas de control supone una transformación geométrica en la que se mapean cuadriláteros irregulares en la trama pasada a cuadrilateros regulares en la trama de predicción. De aquí se puede deducir que a diferencia de los métodos tradicionales de apareamiento de bloques, estos otros métodos no presentan discontinuidades en las fronteras de los cuadriláteros, debido que que el movimiento se modela más bien como una especie de deformación de la malla o rejilla de control en la que además no puede haber zonas de traslape como en los métodos de apareamiento de bloques tradicionales

Respecto a su estructura, esta tesis está dividida en siete capítulos, el primero es la introducción, el segundo habla de los sistemas de compensación de movimiento, y de transformaciones geométricas espaciales muy generales de las que se pueden derivar los diferentes modelos de compensación de movimiento. El tercer capítulo describe las ecuaciones utilizadas para medir el desempeño de los algoritmos. El cuarto capítulo versa sobre los diferentes tipos de algoritmos de apareamiento de bloques, describiendo en forma detallada los algoritmos de búsqueda más comunes empleados por este tipo de esquemas. En el quinto capítulo se introduce el concepto de interpolación bilineal por rejilla de control y se presentan las propuestas desarrolladas como objetivo final de esta tesis. En el capítulo seis se exponen todos los resultados de diferentes combinaciones de pruebas. Y finalmente el capítulo siete concluye este trabajo.

Capítulo 2

Compensación de Movimiento para Compresión de Video Digital

2.1 Introducción

Este capítulo trata primero de la importancia de la compensación de movimiento en sistemas de compresión de video, y explica luego de manera muy general, los principales tipos de algoritmos de compensación.

2.2 Características básicas de un esquema de codificación de video

De acuerdo a las experiencias en la codificación de video a bajas tasas de transmisión, se piensa que las siguientes características son necesarias para un eficiente esquema de codificación de video,

1) Incorporar una operación de compensación de movimiento:

Los cambios intertrama son principalmente provocados ya sea por el movimiento de los objetos, o bien, por el movimiento de la cámara. Si se es capaz de estimar el movimiento que ocasiona el cambio entre tramas y entonces compensarlo, será posible obtener una gran reducción en la tasa de transmisión.

Existen muchos algoritmos de estimación de movimiento muy eficientes. Una excelente revisión de técnicas de estimación de movimiento desarrolladas antes de 1985,

fue expuesta por Musmann *et al* [2]. Desde entonces, se han hecho grandes progresos en estimación de movimiento.

2) Combinación de diferentes esquemas de codificación:

Para comprimir la señal de video a muy bajas tasas de datos, se deben de combinar dos o más esquemas de codificación, puesto que cada esquema tiene sus propias ventajas y desventajas. De esta forma, un esquema apropiado puede operar en su propio dominio favorable. Esta es la razón por la cual los esquemas de compresión eficientes normalmente toman una estructura híbrida.

2.3 Las dos vertientes tradicionales de compensación de movimiento

Como es sabido, la estimación de movimiento está cercanamente relacionada con los tipos de compensación de movimiento empleados. Existen normalmente dos grupos de compensación de movimiento: un conjunto de técnicas utilizan una estimación de movimiento "hacia adelante" en la cual los vectores de movimiento necesitan ser transmitidos. Otro conjunto de técnicas, utiliza una estimación de movimiento recursiva, esto es, los vectores de movimiento se van estimando recursivamente basándose en los pixeles transmitidos. Estos dos grupos de compensación de movimiento especifican dos tipos de algoritmos de estimación de movimiento muy conocidos en la literatura:

1. Algoritmos pel - recursivos, y
2. Algoritmos de apareamiento de bloques

El uso de los algoritmos de apareamiento de bloques, lleva consigo tres características importantes : 1) *La estrategia de búsqueda.* La busqueda exhaustiva o el apareamiento jerárquico se utilizan para codificación a bajas tasas de bits. 2) *El tamaño de los bloques.* Usualmente, 16 x 16 para el formato CIF. 3) *Precisión de los vectores de movimiento.* Para señales de videoteléfono es deseable utilizar precisiones de medio pixel.

Este trabajo de tesis se enfoca a los algoritmos del segundo tipo. En dichos algoritmos, la trama presente debe ser dividida en bloques de pixels, de tal forma que para cada bloque se debe estimar un solo vector de movimiento con respecto a la trama pasada. El flujo de datos transmitido consiste de dos partes: los vectores de movimiento y la señal de actualización de trama (imagen de error). Con el fin de alcanzar una tasa mínima de datos, uno puede desear que el campo de desplazamiento estimado sea óptimo. Un campo de desplazamiento óptimo debe: (1) minimizar la señal residual de movimiento compensado (imagen de error), (2) ser de significado físico real, lo que será

de beneficio para la interpolación de movimiento compensado desarrollada por el receptor.

2.4 Modelando con mejor precisión el campo de desplazamiento.

2.4.1 Modelos de movimiento

Las técnicas de compensación de movimiento utilizadas hoy en día son tadavía de una forma simple, capaces de manejar únicamente movimientos de tipo translacional. Esto conduce a un número de problemas importantes, por ejemplo, efectos de bloque en las imágenes compensadas, o bien, ningún significado físico de los vectores de desplazamiento estimados, etc. Bajo la restricción de "bajas tasas de datos", la obtención de modelos más precisos del campo de desplazamiento se convierte en una tarea cada vez más importante.

En codificación de video a muy bajas tasas de datos, se debe tener un mejor entendimiento del comportamiento del movimiento.. Afortunadamente, las tasas de compresión muy bajas nos dejan más tiempo para emplear modelos de movimiento más avanzados y adoptar técnicas de estimación más sofisticadas.

El movimiento relativo 3-D entre la cámara y los objetos dará por resultado un cambio temporal en el plano de la imagen 2-D. Recordemos que los objetos planos en una secuencia de video son en realidad proyecciones de objetos tridimensionales en movimiento. El cambio temporal 2-D de dichas proyecciones, se puede modelar por modelos de proyección. Existen dos modelos de proyección: *proyección ortogonal y proyección perspectiva* [1].

La proyección ortogonal es una aproximación del proceso de proyección real. En ésta, se aume que todos los "rayos" provenientes del objeto tridimensional viajan paralelos unos a otros hasta el plano donde se forma la imagen [10]. Este tipo de proyección puede ser representada por la transformación *affine*,

$$\begin{aligned} x_1' &= a_1 x_1 + a_2 x_2 + d_1 \\ x_2' &= a_3 x_1 + a_4 x_2 + d_2 \end{aligned} \qquad (2.1)$$

donde (x_1, x_2) son las coordenadas de un pixel en la trama a compensar, y (x_1', x_2') las coordenadas de un punto en la trama de referencia.

En la proyección perspectiva, la imagen se forma usando una "cámara puntual" ideal, de acuerdo a los principios de la óptica geométrica. Todos los "rayos" de un objeto, pasan a través del centro de proyección, el cual está representado por la lente de la cámara. La transformación perspectiva está dada por

$$x_1' = \frac{a_1 x_1 + a_2 x_2 + a_3}{a_7 x_1 + a_8 x_2 + 1}$$

$$x_2' = \frac{a_4 x_1 + a_5 x_2 + a_6}{a_7 x_1 + a_8 x_2 + 1} \qquad (2.2)$$

En capítulos posteriores, se hablará de la transformación bilineal que aunque no está relacionada con ningún movimiento físico en tres dimensiones [10], puede modelar el cambio en la proyección bidimensional tal como lo hace la transformación perspectiva. Esto se debe a que está menos restringida que esta última. Por ejemplo, la transformación perspectiva y la bilineal pueden "mapear" un cuadrado en un cuadrilátero irregular, mientras que sólo la bilineal puede mapear el mismo cuadrado en una figura de cuatro lados no necesariamente rectos. Por otra parte la transformación *affine* "mapeará" el mismo cuadrado en un paralelogramo, necesariamente.

Una vez que el movimiento intertrama ha sido modelado, el problema de la estimación de movimiento se convierte en un problema de identificación de parámetros de movimiento.

2.4.2 Compensación de movimiento

Para un bloque dado $\mathbf{B}_i$ en la imagen de entrada, la tarea de la compensación de movimiento es encontrar el bloque correspondiente $\mathbf{D}_k$ en la imagen de referencia (trama anterior) con el objeto de predecir el bloque $\mathbf{B}_i$ usando $\mathbf{D}_k$, ó bien alineando el bloque $\mathbf{B}_i$ utilizando $\mathbf{D}_k$.

Por conveniencia, se puede seguir la notación utilizada en codificación fractal de imágenes, entonces, $\mathbf{B}_i$ es llamada bloque rango, mientras $\mathbf{D}_k$ es llamada bloque dominio, como se muestra en la figura 2.1.

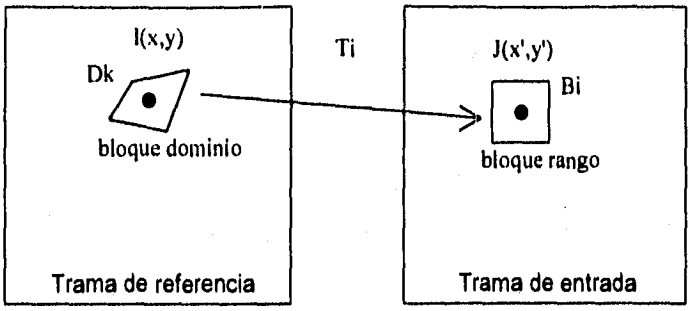


Figura 2.1. El principio de la compensación de movimiento

De esta forma, varias operaciones de compensación de movimiento pueden ser unificadas como,

$$d(B_i, T_i(D_k)) \longrightarrow 0 \tag{2.3}$$

donde *d* es una medida de distancia, normalmente el error cuadrático medio. La transformación $\mathbf{T_i}$ a usar, está definida como

$$\begin{bmatrix} x' \\ y' \\ J(x',y') \end{bmatrix} = T_i \left(\begin{bmatrix} x \\ y \\ I(x,y) \end{bmatrix} \right) \tag{2.4}$$

donde $\{x, y, I(x,y)\} \in D_k$ y $\{x', y', J(x',y')\} \in B_i$. *I(x,y)* representa la luminancia en el bloque dominio.

Por lo tanto, la tarea de la compensación de movimiento es precisamente encontrar la transformación óptima T_i y el bloque dominio $\mathbf{D_k}$ que minimice el criterio de búsqueda anterior.

La transformación T_i es una composición de una transformación geométrica G_i y una transformación másica M_i. Esto se puede representar como

$$T_i = M_i \circ G_i \tag{2.5}$$

a) Compensación de movimiento geométrica: La transformación geométrica G_i determina como mapear un bloque de dominio en un bloque rango. Es utilizada para compensar el desplazamiento del contenido de la escena, provocado por el movimiento relativo entre la cámara y los objetos. Esta se puede describir como

$$\begin{bmatrix} x' \\ y' \\ J(x',y') \end{bmatrix} = G_i\left(\begin{bmatrix} x \\ y \\ I(x,y) \end{bmatrix}\right) = \begin{bmatrix} g_i\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) \\ I(x,y) \end{bmatrix} \tag{2.6}$$

Usualmente, g_i especifica una transformación geométrica espacial. Se utiliza comunmente alguna función polinomial como las que se presentaron en la sección (2.4.2). Las transformaciones polinomiales de órden cero corresponden a un corrimiento de un bloque en la imagen de referencia, y ésta es precisamente la transformación adoptada por los esquemas convencionales de compensación de movimiento basados en bloques (como el método de apareamiento de bloques).

La transformación polinomial de primer orden (esto es, una transformación *affine*), es capaz de manejar un movimiento más general que el solo movimiento translacional. Puede ser valioso notar que una transformación geométrica puede dar como resultado una deformación de los bloques dominio o de los bloques rango. Por ejemplo, un bloque rango rectangular puede quizá corresponder a un bloque dominio cuadrilátero no regular.

b) Compensación de movimiento másica: La transformación másica M_i procesa sólo la imagen en el bloque dominio, la cual afecta los valores de pixel del bloque rango. Una transformación másica se puede describir como

$$\begin{bmatrix} x' \\ y' \\ J(x',y') \end{bmatrix} = M_i\left(\begin{bmatrix} x \\ y \\ I(x,y) \end{bmatrix}\right) = \begin{bmatrix} x \\ y \\ p_i(x,y) + h_i(I(x,y)) \end{bmatrix} \tag{2.7}$$

donde $h_i(.)$ controla el contraste de la imagen y $p_i(.)$ especifica la luminancia de la imagen. Una transformación másica se utiliza principalmente para compensar efectos de luminancia. Por ejemplo, aquellos provocados por el movimiento relativo de un objeto con respecto a la fuente de luz que lo ilumina.

c) Compensación de movimiento transformada: La compensación de movimiento que utiliza la transformación (3) se denomina compensación de movimiento transformada. Especificamente

$$\begin{bmatrix} x' \\ y' \\ J(x',y') \end{bmatrix} = T_i\left(\begin{bmatrix} x \\ y \\ I(x,y) \end{bmatrix}\right) = \begin{bmatrix} g_i\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) \\ p_i(x,y) + h_i(I(x,y)) \end{bmatrix} \tag{2.8}$$

Un esquema de compensación de movimiento transformada, en general supera en desempeño a los esquemas de compensación de movimiento basados en bloques.

Una característica común de varios esquemas de compensación de movimiento como los arriba mencionados, es que la trama a ser predecida es partida en muchos "parches", ya sea regulares o irregulares, y que los "parches" correspondientes en la trama a ser usada para la predicción (trama de referencia), se deben calcular.

De acuerdo a la elección del bloque dominio D_i, la compensación de movimiento puede también ser dividida en grupos: Si a los "parches" correspondientes se les permite traslaparse (como sucede en el esquema de compensación de movimiento basada en bloques, utilizado en esquemas de compresión de video contemporáneos), llamaremos a este esquema, *compensación de movimiento con traslape*. En contraste, si a los "parches" se les prohíbe traslaparse, lo llamaremos *compensación de movimiento sin traslape*. La idea básica de este esquema, es que el movimiento intertrama es tratado como un tipo de *deformación de trama*.

La deformación de trama se puede llevar a cabo ajustando los nodos de rejilla de la malla que definen los "parches", en vez de mover los propios parches.

Una vez que se determinen los desplazamientos de los nodos de la rejilla, el movimiento de los pixels dentro de cada "parche" puede ser obtenido por interpolación. El campo de desplazamiento generado es continuo, lo cual reduce en gran medida al efecto de bloque.

La figura 2.2 muestra una comparación entre estos dos tipos de esquemas. Existen dos tipos de "parches" que se han utilizado para la compensación de movimiento sin traslape: *compensación de movimiento basada en cuadrángulos* (QBMC, quadrangle-based motion compensation) *y compensación de movimiento basada en triángulos* (TBMC, triangle-based motion compesation) [1].

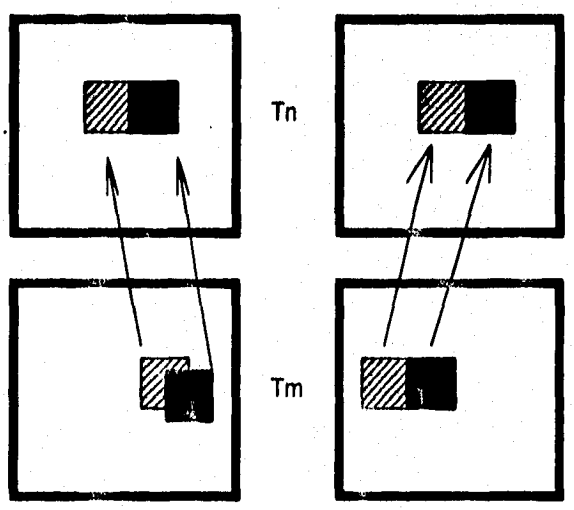


Figura 2.2 Dos tipos de compensación de movimiento

No importando que tipo de "parche" sea utilizado, este tipo de compensación de movimiento consiste de tres pasos a seguir:

i) Se construye una malla de "parches". El primer paso es emplear una malla predeterminada para dividir la imagen de entrada en "parches" pequeños y regulares. Esta técnica se denomina compensación de movimiento por malla fija. Las mallas fijas comunmente utilizadas son la triangular y la cuadrangular.

ii) Se estiman los desplazamientos de los nodos de la rejilla (vértices de los "parches"). La clave de estas técnicas es cómo calcular los desplazamientos de los nodos de la rejilla. Cuando se utilizan esquemas de compensación de movimiento basados en mallas fijas, se debe elejir algún esquema de estimación de desplazamiento. G. Sullivan *et al.* [4] desarrollaron un algoritmo de interpolación de rejilla de control tri-bilineal para estimar el corrimiento de los nodos de rejilla de una malla cuadrangular. Y. Nakaya, presentó un algoritmo de apareamiento hexagonal.

iii) Compensación de movimiento basada en una malla. Una vez que los desplazamientos de los nodos de la rejilla son obtenidos, se puede lograr la compensación de movimiento requerida utilizando técnicas de "mapeo de textura".

Capítulo 3

Medidas de Desempeño

3.1 Introducción

En este capítulo se presentan las expresiones básicas con que se medirá el desempeño de los algoritmos de compensación de movimiento desarrollados en este trabajo de tesis.

3.2 Medidas de fidelidad

Las técnicas comunmente empleadas para compresión de datos de imágenes tráen como resultado una degradación de la imagen reconstruída. Una medida ampliamente utilizada de la fidelidad de la imagen reconstruída para una imagen de tamaño NxM es el error cuadrático medio, el cual se define como

$$e_{ms}^2 = E\left\{(u_{i,j} - u_{i,j}^*)^2\right\} \tag{3.1}$$

donde E representa el operador "esperanza matemática" y $\{u_{i,j}\}$, $\{u^*_{i,j}\}$ representan muestras de las imágenes de NxM original y reconstruida, respectivamente. Experimentalmente, el error cuadrático medio está definido como

$$e_{ms}^2 \cong \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}(u_{i,j} - u_{i,j}^*)^2 \tag{3.2}$$

Existen dos definiciones de la "Relación Señal a Ruido" (SNR) que se utilizan y que corresponden al error antes mencionado. Estas se definen como

SNR =

$$10 \log_{10} \frac{(\text{valor pico a pico de los datos de la imagen original})^2}{e_{ms}^2} (\text{dB}) \qquad (3.3)$$

$$\text{SNR}' = 10 \log_{10} \frac{\sigma_u^2}{e_{ms}^2} (\text{dB}) \qquad (3.4)$$

donde σ_u^2 es la varianza de la imagen original.

Aunque la SNR' se utiliza más ampliamente como una medida de relación señal a ruido en la literatura de procesamiento de señales (puesto que está relacionada con la potencia de la señal y la potencia del ruido), y es quizá más significativa porque dá 0 dB para igual potencia de la señal y del ruido, la SNR es más común en el campo de la codificación de imágenes. Frecuentemente, los datos brutos de la imagen original son dados como muestras discretas cuantizadas a un número relativamente grande de niveles de gris. Típicamente, el número de niveles es 256 (u 8 bits) tal que el valor pico a pico es 255. Por tanto,

$$\text{SNR} = 10 \log_{10} \frac{(255)^2}{e_{ms}^2} \qquad (3.5)$$

A ésta de le llama *relación señal a ruido pico* (PSNR)

3.3 Medidas de cantidad de información

Las tasas de datos de imágenes sin formato no necesariamente representan tasas de información. Por ejemplo, las imágenes monocromáticas se cuantizan típicamente en 8 bits, dando una tasa de datos de 8 bits/pel. Mientras tanto, de acuerdo a la teoría de la información, la tasa de información promedio se dá por la entropía (en bits),

$$H(u) = -\sum_{i=1}^{L} p_i \log_2 p_i \qquad (3.6)$$

donde p_i es la probabilidad de que una muestra cuantizada u tome el valor i, de un conjunto de $L = 2^n$ valores (n=8, aquí).

Veamos con detalle de manera formal el porqué de esta expresión de información.

Una fuente X con un alfabeto A está definida como un proceso aleatorio discreto (una secuencia de variables aleatorias X_i, $i = 1, ...$) en la forma $X = X_1\ X_2\ ....$, donde cada variable aleatoria X_i toma un valor del alfabeto A. En el futuro, asumiremos que el alfabeto contiene un número finito (M) de símbolos, es decir, $A = \{ a_1, a_2, ... , a_M \}$. Llamemos DMS a una fuente discreta sin memoria, la cual es tal que los símbolos sucesivos son estadísticamente independientes. Esta está completamente especificada por las probabilidades $p(a_i) = p_i,\ i = 1,.....,M$ tales que $p_1 + ... + p_M = 1$.

De acuerdo con la teoría de la información, el contenido de información de un símbolo depende de incertidumbre, es decir, de que tan impredecible o inesperado es. Si un símbolo con baja probabilidad ocurre, una cantidad mayor de información se tiene que transmitir, que cuando se dá la ocurrencia de un símbolo más probable, Este concepto cuantitativo de "sorpresa" está formalmente expresado por la relación

$$I(a_i) = \log_2(1 / p(a_i)), \quad \text{para } a_i \in A \tag{3.7}$$

donde $I(a_i)$ es la cantidad de información que el símbolo a_i con probabilidad $p(a_i)$, lleva. La unidad de información es el "bit" cuando se usa el logaritmo en base 2. Obsérvese que si $p=1$, entonces $I = 0$ como es esperado, y conforme $p \to 0$, $I \to \infty$ En la práctica, la probabilidad de ocurrencia de cada símbolo se estima del histograma de una fuente específica, como el caso del histograma de niveles de gris de una imagen, o bien, como se utilizará posteriormente: *el histograma de valores del campo vectorial de desplazamiento* utilizado para compensación de movimiento.

La entropía H(X) de una fuente X, DMS con un alfabeto A, se define como la información promedio por símbolo en la fuente, dada por

$$H(X) = \sum_{a \in A} p(a) \log_2(1 / p(a))$$

$$= -\sum_{a \in A} p(a) \log_2 p(a) \tag{3.8}$$

Mientras más sesgada esté la distribución de probabilidad de los símbolos, más pequeña será la entropía de la fuente. La entropía se ve maximizada cuando la distribución es plana, esto es, cuando todos los símbolos son igualmente probables. De esto último, se sigue que una fuente donde algunos símbolos son más probables que otros, tiene una entropía más pequeña que una fuente en la que todos los símbolos son igualmente probables.

A manera de ejemplo, consideremos el caso de la entropía de los datos crudos de una imagen de 256 niveles de gris, y el caso de la entropía de los vectores de un campo de desplazamiento obtenido mediante un BMA (Algoritmo de Apareamiento de Bloques).

Ejemplo 1: Entropía de datos de una imagen sin formato.

Supóngase que una imágen de 8 bits es tomada como una realización de una fuente X tipo DMS. Los símbolos i son los niveles de gris de los pixels, y el alfabeto A es la colección de todos los niveles de gris entre 0 y 255. Entonces la entropía de la imagen está dada por

$$H(X) = -\sum_{i=0}^{255} p(i)\log_2 p(i)$$

donde $p(i)$ denota la frecuencia relativa de ocurrencia del nivel de gris i en la imagen. Nótese que la entropía de una imágen consistente de un solo nivel de gris (imagen constante) es igual a cero.

Ejemplo 2: Entropía de un campo de desplazamiento.

Supóngase ahora, que con dos tramas de 256x256 pixels, de una secuencia de video, se efectúa un BMA tradicional por búsqueda exhaustiva con bloques de 8x8, y se supone un desplazamiento máximo de pixel intertrama de $p = 11$. Entonces, se tendrá un total de 32x32 vectores de movimiento para cada uno de los bloques que dividen la imagen. Por otro lado, cada vector podrá tener un valor de entre (2p+1)x(2p+1) = (2x(11)+1) x (2x(11)+1) = 529 posibilidades. Así, la entropía estará dada por

$$H(X) = -\sum_{m=-p}^{p}\sum_{n=-p}^{p} p(V_{m,n})\log_2 p(V_{m,n})$$

donde $p(V_{m,n})$ denota la frecuencia de ocurrencia de un vector $V_{m,n}$ en el campo de desplazamiento. De esta forma, la entropía dará el número mínimo promedio de bits por símbolo (es decir, de bits por vector) necesarios para la codificación entrópica.

En este ejemplo, la entropía de un campo de vectores de movimiento, estará definida especificamente por,

$$\text{Entropía del CampoVectorial} = \frac{1}{N_t}\sum_{m=-p}^{p}\sum_{n=-p}^{p} N_{m,n}\log_2\left(\frac{N_t}{N_{m,n}}\right) \quad \text{bits / vector}$$

donde $N_{m,n}$ es el número de veces que un vector de movimiento con valor (m,n) es utilizado en la trama, y N_t es el número total de vectores de movimiento en la trama.

En 1948, Shannon definió el teorema de la codificación sin pérdidas. La tasa de datos mínima que puede ser lograda por medio de la codificación sin pérdidas de una fuente X discreta sin memoria, está dada por

$$\min\{R\} = H(X) + \varepsilon \qquad \text{bits / símbolo} \tag{3.9}$$

donde R es la tasa de transmisión, H(X) es la entropía de la fuente, y ε es una cantidad positiva que puede ser arbitrariamente cercana a cero.

El teorema de la codificación sin pérdidas establece el límite inferior para la tasa de datos necesaria para alcanzar un error de codificación y decodificación de cero. En el caso de un DMS, podemos acercarnos a este límite codificando cada símbolo independientemente. Una forma para hacerlo es utilizar una técnica de codificación entrópica como la codificación Huffman [9], con la cual uno puede aproximarse a este límite.

Capítulo 4

Compensación de Movimiento por Apareamiento de Bloques

4.1 Introducción

El apareamiento de bloques se puede considerar como el método más difundido para la estimación de movimiento, debido a la poca complejidad que se necesita en el desarrollo del "hardware". A consecuencia de lo último, resulta ampliamente disponible en VLSI, y casi todos los codificadores H.261 y MPEG 1-2 utilizan el apareamiento de bloquesm, para la estimación de movimiento [10].

El apareamiento de bloques es un método que busca la mejor estimación del vector de movimiento por medio de un procedimiento de búsqueda en el dominio de los pixels , es decir, en el dominio de la imagen.

En este capítulo se desarrolla el principio básico de los métodos que utilizan apareamiento de bloques para realizar la compensación de movimiento, y se detallan algunos de los métodos de búsqueda más conocidos utilizados por los algoritmos de apareamiento de bloques (BMA's).

4.2 Premisas básicas para la estimación de movimiento por bloques

El movimiento entre tramas ("*interframe*", en inglés) se puede modelar de manera aproximada por translaciones lineales de "piezas" o bloques constitutivos de los objetos que se mueven.

Si una escena de televisión contiene objetos en movimiento y si es factible de obtenerse un estimado de su translación, entonces, se podrá realizar una predicción mucho más eficiente utilizando elementos de la trama previa que estén apropiadamente desplazados (espacialmente).

En escenas reales de televisión, el movimiento de los objetos puede ser una complicada combinación de translaciones, rotaciones, acercamientos, deformaciones ,etc..

De los tipos de movimiento mencionados, el movimiento translacional es relativamente fácil de estimar, y de hecho, los algoritmos tradicionales que se utilizan para compensación de movimiento parten de la premisa de que los objetos en la escena sólo se mueven translacionalmente, lo que implica una muy aproximada estimación del movimiento. El éxito de estas técnicas se basa entonces, en la gran cantidad de movimiento translacional que existe en la escena.

Como se puede consultar en [9], la mayoría de los algoritmos para estimación de movimiento en codificación "intertrama" basan su operación en las siguientes suposiciones:

(i) Los objetos se mueven en translación en un plano que es paralelo al plano de la cámara. No se consideran efectos de acercamiento ("zoom"), ni de rotación.

(ii) La iluminación es uniforme temporal y espacialmente.

(iii) La oclusión de un objeto por otro y el fondo descubierto no se toman en cuenta.

De estas tres premisas se obtiene el fundamento más importante de los algoritmos de estimación de movimiento:

Cada pixel de una trama puede ser encontrado en la siguiente, en la misma u otra posición pero su intensidad permanece sin alteración,

$$u_k(x,y) = u_{k-1}(x-q,y-l) \tag{4.1}$$

donde k es la trama presente y $k-1$ la trama anterior. $\mathbf{D} = (q,l)$ es el vector de translación bidimensional del pixel durante el intervalo de tiempo $(k-1, k)$, y (x,y) es el vector

bidimensional de la posición espacial. El problema es entonces, estimar **D** a partir de las intensidades de los pixels de las tramas presente y pasada.

4.3 El método de apareamiento de bloques para estimación de movimiento

La idea básica del apareamiento de bloques se ilustra en la figura 4.1, donde el desplazamiento para un pixel (n_1,n_2) en la trama k (la trama presente) se determina considerando un bloque de NxN centrado alrededor de (n_1,n_2), y buscando la posición de su su mejor acoplamiento sobre un bloque del mismo tamaño en la trama de búsqueda, que en este caso es la trama k+1.

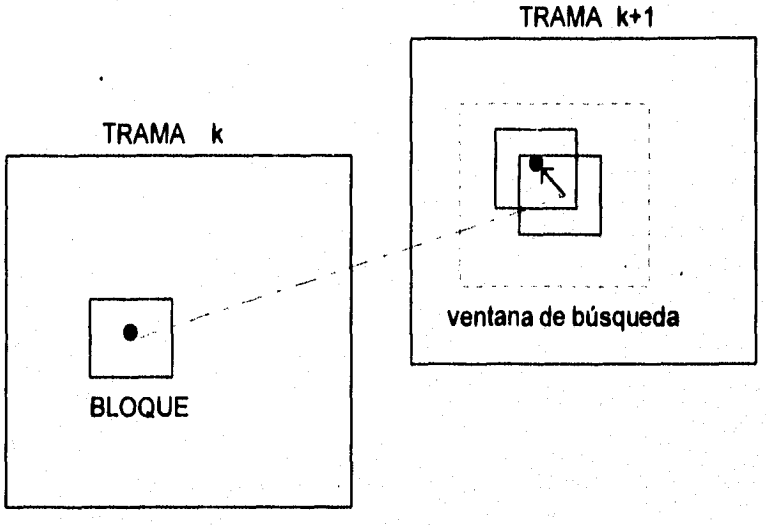


Figura 4.1 Apareamiento de bloques para estimación de movimiento

La búsqueda está usualmente limitada a una región de tamaño $(2d_{máx} + N)x(2d_{máx}+N)$, donde $d_{máx}$ representa el desplazamiento horizontal o vertical máximo entre tramas.

A esta región se le denomina comunmente *ventana de búsqueda*, por razones computacionales.

Los BMA's difieren en tres características esenciales:

- El criterio de apareamiento.
- La estrategia de búsqueda.
- La determinación del tamaño de bloque.

4.3.1 Criterios de apareamiento

El apareamiento de bloques se puede cuantificar de acuerdo a varios criterios, tales como el error cuadrático medio mínimo (MSE mínimo), la diferencia absoluta media mínima (MAD mínimo), y la cuenta de pels de apareamiento máxima (MPC).

Para el MSE mínimo, se evalúa el MSE definido como

$$MSE(i,j) = \frac{1}{MN} \sum_{(m,n \in B)} \left[s_k(m,n) - s_{k+1}(m+i,n+j) \right]^2 \tag{4.2}$$

donde B denota un bloque de MxN, para un conjunto de vectores de movimiento candidatos (i,j). El estimado del vector de movimiento será el valor (i,j) que minimice el criterio MSE. Esto es,

$$(\hat{i},\hat{j}) = \underset{(i,j)}{\operatorname{argmin}} \quad MSE(i,j) \tag{4.3}$$

El criterio del mínimo MSE comunmente no se utiliza en implantaciones VLSI, esto debido a que resulta muy complicado realizar la operación de "elevar al cuadrado" en "hardware".

En cambio, el criterio del mínimo MAD, definido como

$$MAD(i,j) = \frac{1}{MN} \sum_{(m,n) \in B} \left| s_k(m,n) - s_{k+1}(m+i,n+j) \right| \tag{4.4}$$

es la elección más usada en las aplicaciones VLSI. Así, el estimado del desplazamiento está dado por

$$(\hat{i},\hat{j}) = \underset{(i,j)}{\operatorname{argmin}} \quad MAD(i,j) \tag{4.5}$$

Es conocido que el desempeño del criterio MAD se deteriora conforme el área de búsqueda se vuelve muy grande debido a la presencia de muchos mínimos locales [10].

Otra alternativa es el criterio de "cuenta máxima de pels de apareamiento". En este esquema cada pixel dentro del bloque B se clasifica, ya sea como un pixel de apareamiento, o bien, como un pixel de desacoplo de acuerdo a

$$T(m,n;i,j) = \begin{cases} 1 & si \left| s_k(m,n) - s_{k+1}(m+i,n+j) \right| \le t \\ 0 & de \quad otra \quad manera \end{cases} \tag{4.6}$$

donde t es un umbral predeterminado. De esta manera, el número de pixels de apareamiento dentro del bloque B está dado por

$$MCP(i,j) = \sum_{(m,n)\in B} T(m,n;i,j) \tag{4.7}$$

Y el vector estimado, será

$$(\hat{i},\hat{j}) = \underset{(i,j)}{\text{argmax}} \quad MPC(i,j) \tag{4.8}$$

4.3.2 Estrategias de búsqueda

Si se desea estimar el desplazamiento de un pixel de coordenadas (x,y) de la trama presente (trama k) a la trama de búsqueda (trama k+1), centramos un bloque de MxN en dicho pixel en la trama presente. Si suponemos que sólo puede haber desplazamientos máximos horizontales y verticales de $d_{máx}$ pixels, entonces, la región en la trama de búsqueda donde se debe evaluar el criterio de apareamiento, debe ser una región de $(M+2d_{max})x(N+2d_{max})$ pixels centrada en (x,y). Esta sería la región de búsqueda. Y el criterio de apareamiento se evaluaría en $(2d_{máx} + 1)x(2d_{máx} + 1)$ posiciones del bloque dentro de la región de búsqueda.

Nótese que aquí hablamos de una trama presente (k) y una trama de búsqueda que hasta este momento se ha supuesto que es la trama k+1. Si embargo, más adelante veremos que en el caso de los esquemas de compensación de movimiento para compresión de video, la trama de búsqueda suele ser la trama pasada k-1, y además se considera que el vector obtenido al final de la búsqueda corresponde no sólo al pixel central del bloque sino a todos los pixels dentro del bloque.

Si por ejemplo, el tamaño del bloque es de 8x8 y el máximo desplazamiento posible es $d_{max}= 10$, la región de búsqueda en la trama de búsqueda será una región de 28x28 pixels.

Método de la búsqueda exhaustiva

La búsqueda eshaustiva es la manera más simple de encontrar la posición (i,j) donde el criterio de apareamiento se hace mínimo. Este método consiste en evaluar dicho criterio en las $(2d_{máx} + 1)x(2d_{máx} + 1)$ posiciones del bloque dentro de la ventana de búsqueda, y encontrar la posición (i,j) donde el MSE o el MAD sea mínimo. $(\hat{i},\hat{j})$ será entonces el vector de desplazamiento estimado. Este método es por tanto muy pesado computacionalmente hablando.

Se han propuesto varios métodos para realizar la búsqueda sin tener que evaluar el criterio de apareamiento sobre toda el área de búsqueda.

Con la meta de superar esta dificultad, se han investigado varios métodos que simplifican el procedimiento de búsqueda.

Algoritmo de búsqueda logarítmica bidimensional (2-D).

J.R. Jain y A.K. Jain en 1981 (véase [8]), sugirieron un método para reducir el número de búsquedas requeridas para encontrar el mejor acoplamiento.

El procedimiento de búsqueda logarítmica bidimensional [2], se basa en la suposición de que el criterio de apareamiento MSE (también conocido como criterio de distorsión [2]), se incrementa monotonamente a medida que la búsqueda se aleja de la dirección de mínima distorsión. La dirección de mínima distorsión está definida por (i,j), tal que *Distorsión(i,j)=MSE(i,j)* sea mínima.

En este método, cinco puntos de búsqueda son analizados, como se muestra en la figura 4.2. La distancia entre los puntos de búsqueda debe ser reducida si el mínimo está en el centro de los puntos de búsqueda o en la frontera del área de búsqueda.

El procedimiento continúa hasta que el plano de búsqueda se reduce a un tamaño de 3x3. En el paso final, se realiza la búsqueda en las nueve posiciones, y la que tenga el mínimo MSE será la posición $(\hat{i}, \hat{j})$ final.

En el ejemplo de la figura 4.2, se requiren cinco pasos para encontrar el vector de desplazamiento en el punto (2,6).

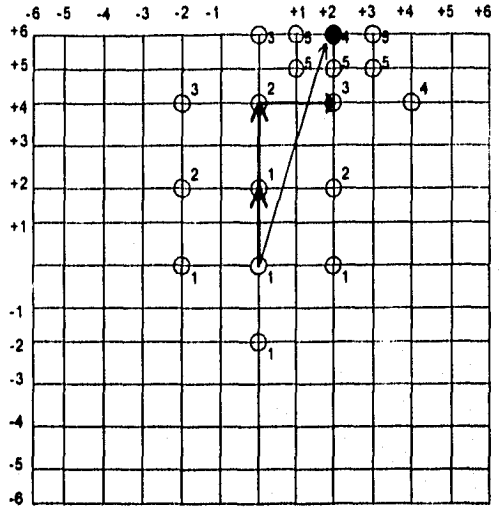


Figura 4.2 Procedimiento de búsqueda logarítmica en 2-D.

Los puntos de la gráfica representan las posiciones posibles que puede tener el bloque dentro del área de búsqueda. El número total de posiciones en que se puede encontrar el bloque dentro del área o ventana de búsqueda es $(2d_{máx} + 1)x(2d_{máx} + 1)$. Por otra parte, el área de búsqueda es un rectángulo que representa el área de movilidad del bloque, y su dimensión es de $(M+2d_{max})x(N+2d_{max})$.

Formalmente el algoritmo consta de los siguientes pasos:

Para un entero $m>0$, se define

$$(m) = \{(i,j);\quad -m \le i,j \le m\}$$

$$(m) = \{(0,0),(m,0),(0,m),(-m,0),(0,-m)\},$$

Algoritmo Log 2-D:

PASO 1:

$$(i,j) = \infty \qquad (i,j) \notin N(d_{máx})$$

$$n' = \lfloor \log_2 d_{máx} \rfloor$$

$$n = \max\{2, 2^{n'-1}\}$$

donde $\lfloor \bullet \rfloor$ es la función de truncatura de entero más pequeño.

PASO 2: $M'(n) \leftarrow M(n)$

PASO 3: *Encontrar* $(i,j) \in M'(n)$ *tal que* $MSE(i+\hat{i}, j+\hat{j})$ *sea mínimo. Si i=0 y j=0, ir al paso5; de otra manera ir al paso 4.*

PASO 4: $\hat{i} \leftarrow \hat{i}+i, \quad \hat{j} \leftarrow \hat{j}+j; \quad M'(n) \leftarrow M'(n)-(-i,-j);$ ir *al paso 3.*

PASO 5: $n \leftarrow n/2$. *Si* $n=1$ *ir al paso 6; de otra manera, ir la paso 2.*

PASO 6: *Encuentre* $(i,j) \in N(1)$ tal que $MSE(i+\hat{i}, j+\hat{j})$ *es mínimo.* $\hat{i} \leftarrow i+\hat{i}, \quad \hat{j} \leftarrow \hat{j}+j.$ $(\hat{i}, \hat{j})$ *da el punto de mínima distorsión (MSE mínimo).*

La figura 4.2 ilustra el procedimiento de búsqueda para $d_{máx} = 5$.

Algoritmo de búsqueda en tres pasos

Koga et al. propusieron un procedimiento de búsqueda de "tres pasos" [2]. En este, esquema un "0" marca el pixel en la trama de búsqueda que está justo en la posición del pixel presente. En el primer paso, el criterio de apareamiento se evalúa en nueve puntos que son el pixel "0" y los pixels marcados como "1". Si el MAD más bajo es encontrado en el pixel "0", entonces se tiene "movimiento nulo". En un segundo paso, ocho puntos de búsqueda marcados como "2" que están espaciados más cercanamente alrededor de la primera aproximación (marcada como "1") se prueban para la mínima distorsión, de forma tal que el punto (i+3,j+5) es encontrado para este ejemplo. Se repite el segundo paso hasta que se alcance la precisión requerida. Para el caso particular de un área de búsqueda con $d_{max} \leq 6$, el tercer paso proporciona el vector de desplazamiento final que en este ejemplo es (i+2,j+6).

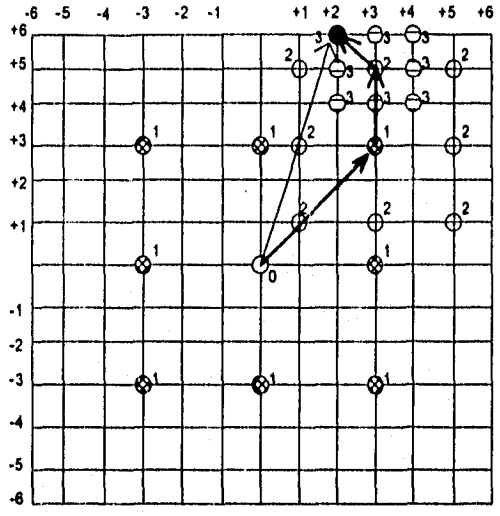


Figura 4.3. Procedimiento de búsqueda en tres pasos.

Algoritmo de búsqueda en la dirección conjugada

Srinivasan y Rao (véase [6]) presentaron un algoritmo de búsqueda llamado "búsqueda en la dirección conjugada", el cual modificado después y con el nombre de "búsqueda de uno a la vez ", ofrece mejores resultados que el primero.

METODOLOGIA BASICA

Los métodos de búsqueda para optimización, intentan reducir el valor de alguna función objetiva mediante el uso de pruebas ("tests") cerca de una estimada de la solución. Se ve una dirección de búsqueda en la cual se espera encontrar el mínimo. Puesto que la dirección obtenida puede no ser correcta, el proceso es iterativo. Después de cada paso de minimización, búsquedas subsecuentes son efectuadas hasta que se satisface un criterio que define el momento en que el mínimo deseado se ha encontrado.

Una técnica simple en una optimización de dos variables (como en nuestro caso) es ajustar una variable a la vez para mínima distorsión. Esto se repite hasta que ya no se pueda alcanzar ninguna mejora.

Tal tipo de ajuste es denominado *búsqueda de uno a la vez* (OTS: One at the time search). Este es un método de búsqueda básico. En un caso de optimización de dos variables, cada variable se ajusta con la otra fija en esos momentos. La dirección de búsqueda en cada paso será paralela a uno de los ejes coordenados. La figura 4.4 ilustra la OTS para una función convexa de dos variables.

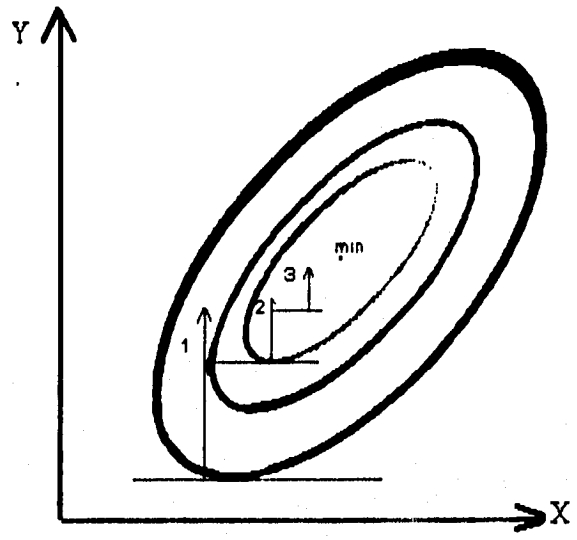


Figura 4.4 Método de búsqueda "uno a la vez"

El método de búsqueda OTS puede ser simplificado cuando la ventana de búsqueda no es muy grande, cambiando sólo una vez la dirección de búsqueda. Es decir, primero buscamos un mínimo en la dirección horizontal *x* y luego (habiendo fijado x) buscamos un mínimo en la dirección vertical *y*. Cuando se halle este último, el algorítmo termina sin regresar más a la dirección horizontal.

A esta variante simplificada del método OTS se le conoce como "búsqueda en la dirección conjugada" y se ilustra en la figura 4.5.

Este algoritmo usa el criterio MAD. En la primera búsqueda, se determina el mínimo en la dirección *i* calculando D(i-1,j), D(i,j), D(i+1,j), donde D representa el error obtenido. Si D(i+1,j) resulta ser la menor, entonces D(i+2,j) también debe ser calculado.

Si se continúa de esta forma, el mínimo en la dirección *i* se detecta cuando el valor más pequeño está situado entre dos valores más altos.

Posteriormente en una búsqueda subsecuente, el mínimo en la dirección j se determina mediante el mismo procedimiento, comenzando en el mínimo de la primera búsqueda.

La figura 4.5 ilustra el método de "búsqueda en la dirección conjugada". Este método por su sencillez resulta ser el más apropiado para implantarse en "hardware", además de que ofrece una mucho mayor rapidez de ejecución debido a que realiza pocas búsquedas en comparación con los métodos anteriores.

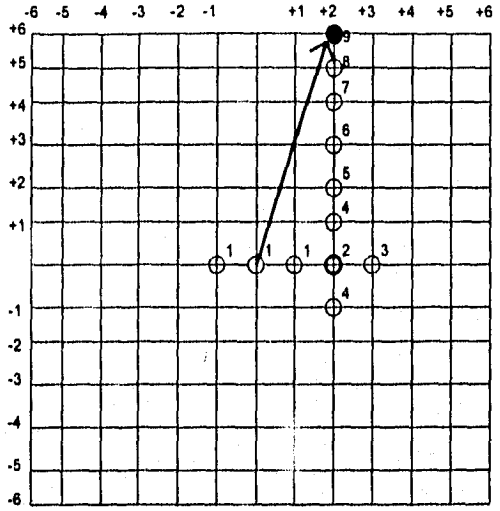


Figura 4.5. Búsqueda en la dirección conjugada.

Algoritmo OTS modificado

Como parte de este trabajo de tesis se propuso un algoritmo OTS modificado de sencilla implementación que podemos resumir en el siguiente seudocódigo.

PASO 1: Se calcula el criterio de apareamiento o distorsión en el pixel inicial MAD(i,j)
PASO 2 Se inicializan las banderas : STOP = 0, *izq_der* = 1, *arr_abj* = 1.
PASO 3. Se define un Umbral.
While (STOP=0)

```
{
Si D(i, j+_izq_der)<D(i,j)
        {
         i = i,
         j= j + izq_der,
        }
  else
        {
        si  izq_der=1              {izq_der= - 1}
        si  izq_der= - 1   {izq_der= + 1}
         i = i
         j = j
        }
Si D(i+arr_abj, j)<D(i,j)
        {
        i = i + arr_abj,
        j = j,
        }
  else
        {
        si  arr_abj=1              {arr_abj= - 1}
        si  arr_abj= - 1   {arr_abj= + 1}
```

```
                    i = i
                    j = j
                    }
          Si D(i,j) < Umbral  {STOP = 0}
          }
Fin del algoritmo.
```

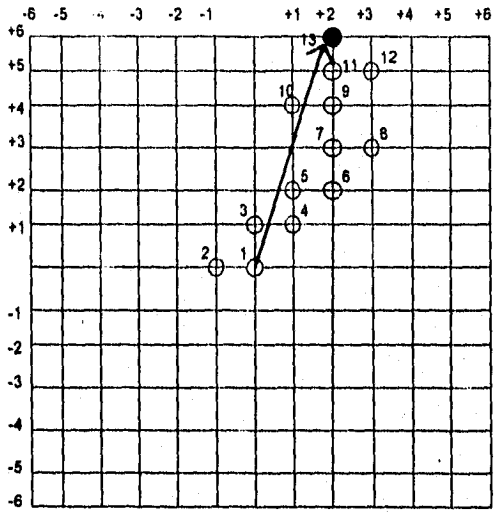


Figura 4.6 Algoritmo OTS modificado

Comparación en el desempeño de los métodos de búsqueda

La rapidez de ejecución computacional de un método de búsqueda se puede medir por el número de búsquedas requeridas. En los ejemplos anteriores, se utilizó un desplazamiento máximo de dmax = 6. Para este caso, el método de "la fuerza bruta" requiere Q=169 puntos de búsqueda. A continuación se muestra una tabla que presenta una comparación del número de puntos de búsqueda y pasos secuenciales que se requieren para los procedimientos de búsqueda explicados anteriormente.

Procedimiento de búsqueda	Número requerido de puntos de búsqueda	Número requerido de pasos secuenciales
logarítmica 2-D	18	5
tres pasos	25	3
Búsqueda conjugada	12	9
OTS modificado	13	13

Para el vector de desplazamiento (i+2,j+6)

Tabla 4.1

Como se puede ver de la tabla 4.1, el algoritmo de la "búsqueda conjugada" es el que efectúa el menor número de pasos de búsqueda, razón por la cual, es el más rápido de todos. Sin embargo, en pruebas experimentales, se comprobó que el método "OTS modificado" ofrece un desempeño casi igual que el anterior en tiempo de ejecución, y proporciona un error de distorsión más bajo.

4.4 El método de apareamiento de bloques para compensación de movimiento en compresión de video

Para reducir la complejidad computacional del procesamiento en tiempo real, una trama de video es dividida en bloques cuadrados de tamaño fijo, con la suposición de que todos los pixels en un bloque siguen el mismo desplazamiento. De tal forma que los criterios de búsqueda mencionados anteriormente (MSE, MAD, etc.) pueden ser adoptados [5].

La tarea de la compensación de movimiento usando BMA's, es formar una imagen compensada o de predicción de la trama presente, en base a una trama de referencia, que generalmente es la trama pasada, y la menor cantidad posible de parámetros de movimiento.

Por tanto, en este esquema, se divide la trama presente en bloques de NxN, y para cada bloque se estima un vector que marca su posición aproximada en la trama anterior. La siguiente figura ilustra la manera más simple que puede adoptar un sistema no híbrido de compresión de video por compensación de movimiento.

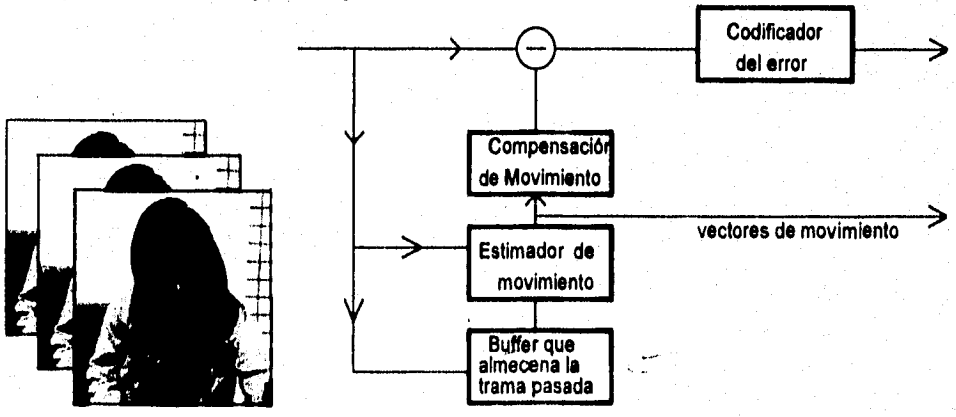


Figura 4.7 Forma simple de un codificador con Compensación de Movimiento

Lo que se busca es que la trama compensada o de predicción sea lo más parecida a la trama actual, de tal forma que la imagen de diferencia entre éstas dos, conocida como imagen de error, contenga una energía mínima, y así pueda ser codificada con el menor

número de bits y enviada al receptor junto con los vectores de movimiento (uno por cada bloque). La figura 4.8 ilustra el decodificador correspondiente.

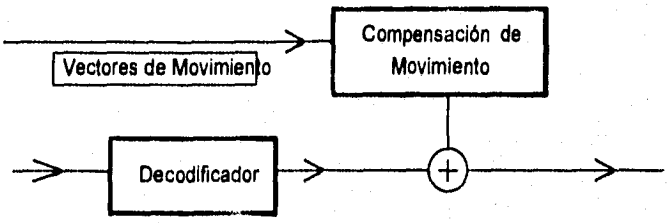


Figura 4.8 Decodificador

La estimación de movimiento por técnicas de apareamiento de bloques, constituye una herramienta muy adecuada y bastante simple, para lograr bajas tasas de bits en transmisión o almacenamiento de secuencias de imágenes con movimiento. En [8] se menciona que la tasa de bits se puede llegar a reducir en un factor de dos utilizando compensación de movimiento.

Capítulo 5

Compensación de Movimiento por Rejillas Deformables

5.1 Introducción

En este capítulo se explican los métodos de compensación de movimiento que utilizan redes o rejillas deformables.

Los métodos dominantes de compensación de movimiento, como el apareamiento de bloques, representan un caso especial de las transformaciones geométricas, como se vió en el capítulo 2.

Un tipo especial de transformación geométrica es precisamente la interpolación por rejilla de control (CGI, *control grid interpolation*) mencionada también anteriormente. Sullivan [4], presentó un método muy efectivo de CGI que produce un campo de desplazamiento suave, y que por tanto, preserva la continuidad y la conectividad en la imagen de predicción.

Si se utiliza compensación de movimiento, los algoritmos de compresión pueden reducir significativamente la cantidad de bits que se necesitan para transmitir imágenes que posean una calidad aceptable.

Los métodos dominantes para compensación de movimiento son los algoritmos de apareamiento de bloques (BMA's). Como ya se ha mencionado, en un BMA la imagen de entrada es descompuesta en una rejilla regular de bloques rectangulares. Un vector de movimiento (MV) se debe encontrar para cada bloque, y todos los pixels del bloque se desplazan de igual forma. Los vectores de movimiento se codifican de entrópicamente y

se envían al receptor. Un método directo pero computacionalmente costoso para encontrar estos vectores de movimiento es la búsqueda exhaustiva.

Existen problemas muy bien conocidos inherentes a los métodos de apareamiento de bloques. Cuando objetos que describen distintos tipos de movimiento se encuentran en el mismo bloque, el hecho de utilizar el mismo vector de movimiento para todos los pixels del bloque constituye un serio error en la compensación.

También, cuando bloques adyacentes utilizan vectores de movimiento muy diferentes, se crea una discontinuidad de "línea recta" en la imagen de predicción. Puesto que el sistema visual humano es altamente sensitivo a las líneas rectas, estos errores aparecen como artefectos de bloque.

Por otro lado, puesto que los vectores de movimiento se determinan independientemente para cada bloque, el campo de vectores de movimiento es discontinuo, difícil de comprimir y puede tener poca relación con el movimiento verdadero de los objetos.

Más aún, puesto que cada bloque puede seguir sólo un movimiento traslacional, los BMA's tienen dificultad para compensar otros tipos de movimiento tales como la rotación y los cambios de profundidad. Estos problemas demuestran la necesidad de construir métodos que operen con pixels individuales, en vez de grandes bloques de ellos.

La interpolación por rejilla de control (que denotamos como CGI) es un método de transformación espacial que puede ser utilizado para crear un campo de desplazamiento interpolado suave.

En [4] Sullivan *et al.*, se muestra que los BMA's son un caso especial de CGI, y se describe un algoritmo CGI que es compatible con casi cualquier sistema de codificación BMA, y que además no produce artefectos de bloque. Utiliza la misma -o inclusive más baja- tasa de transmisión que los BMA's y tiene una distorsión cuadrática media similar o más alta.

5.2 Interpolación por rejilla de control

La interpolación por rejilla de control es una técnica, para desarrollar transformaciones espaciales en una imagen. Comienza con desplazamientos específicos para un pequeño número de puntos en una imagen, llamados puntos o nodos de control.

Los desplazamientos espaciales de todos los demás puntos (o pixels) en la imágen se determinan interpolando entre los desplazamientos de los nodos de control.

Los puntos de control utilizados para esta transformación, se eligen normalmente como los vértices de una rejilla rectangular y regular orientada horizontalmente en la imagen de salida, como se puede apreciar en la figura 5.1.

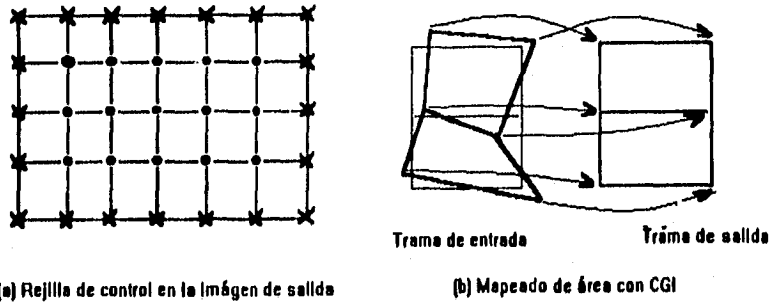


Figura 5.1. Mapeado por interpolación de rejilla de control

Sin embargo, cualquier conjunto, regularmente o arbitrariamente colocado, podría utilizarse. En el caso de una rejilla rectangular, los valores de desplazamiento de los nodos de control, determinan los vértices de un conjunto de cuadriláteros contiguos (probablemente irregulares) en la imagen de entrada y que van a ser mapeados a la rejilla rectangular en la imagen de salida.

Para compensación de movimiento, la imagen de entrada será la trama previa decodificada $\left\{\hat{S}_{k-1}(x,y)\right\}$, y la imagen de salida será la imagen compensada predicha $\left\{\hat{S}_{k-1}(x-\Delta x, y-\Delta y)\right\}$. Se definen también los nodos de control exteriores y los interiores como aquellos en los bordes de la trama y aquellos en el centro de ésta respectivamente, como se ilustra en la figura 5.1

Un caso especial y trivial de CGI, es como ya se ha mencionado, la compensación de movimiento BMA. En ésta, se puede considerar el vector de movimiento para cada bloque como el desplazamiento del centro del bloque. Entonces, el desplazamiento de un pixel puede ser determinado por "interpolación del vecino más cercano" entre los desplazamientos especificados para cada punto de control, es decir, todos los pixels en un bloque utilizan el mismo vector de desplazamiento.

Así, los artefactos de bloque, encontrados en un BMA, son simplemente el resultado de la forma más burda de interpolación ("vecino más cercano").

De la misma forma que una interpolación de orden superior puede remover discontinuidades en formas de onda unidimensionales, puede también remover discontinuidades espaciales en una imagen bidimensional. Un tipo muy simple de interpolación bidimensional que puede desempeñar esta tarea es la *interpolación bilineal.*

5.3 La interpolación bilineal

Una transformación geométrica se compone de dos operaciones básicas: a) una transformación espacial que define el "reacomodo" de los pixels de la imagen original y b) una interpolación que involucra los valores de los pixels originales, con el objeto de generar los valores de los pixels de salida en una rejilla discreta de la imagen resultante.

En cuanto a la interpolación, su forma general se describe como sigue

$$g(i,j) = \sum_{\hat{u}} \sum_{\hat{v}} f(\hat{u},\hat{v}) R(j-\hat{u}, k-\hat{v}), \quad (\hat{u},\hat{v}) \in V_{\hat{u}\hat{v}} \tag{5.1}$$

donde R es la función de interpolación, y $V_{\hat{u}\hat{v}}$ es una subregión finita de la imagen original, es decir, para evaluar el valor de un pixel de salida se toman solamente los pixels de una vecindad finita que contenga al pixel de coordenadas $(\hat{u},\hat{v})$.

En cuanto a la interpolación lineal, que en procesamiento de imágenes se conoce más bien como interpolación bilineal, consideremos la figura 5.2 en la que aparece una vecindad de forma cuadrada ,

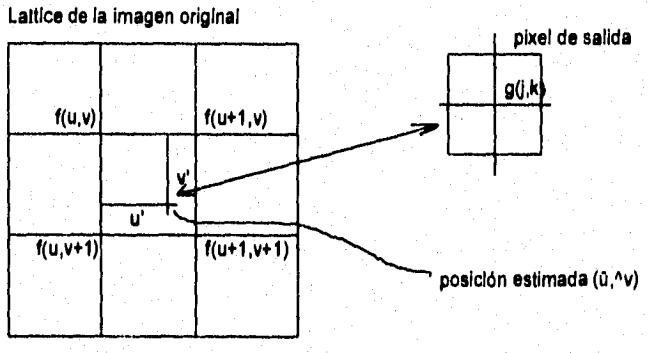


Figura 5.2 Interpolación bilineal

alrededor del pixel cuya posición $(\hat{u},\hat{v})$ ha sido estimada. Entonces, a partir de los pixels más cercanos se obtienen los interpolantes

$$\phi(u',v) = u'f(u+1,v) + (1-u')f(u,v) \tag{5.2}$$

y

$$\phi(u', v+1) = u'f(u+1, v+1) + (1-u')f(u, v+1) \tag{5.3}$$

donde se ha supuestro que de pixel a pixel, en la imagen original, hay una distancia de uno. Nótese además, que las posiciones $\hat{u}$, $\hat{v}$ se miden a partir de (u, v) y son respectivamente u' y v'. El paso final, es ahora interpolar linealmente entre $\phi(u', v)$ y $\phi(u', v+1)$ para obtener

$$g(j,k) = \phi(u', v') = (1-v')\phi(u', v) + v'\phi(u', v+1) \tag{5.4}$$

es decir, g(j,k) es el valor interpolado del pixel de salida cuya posición es (j,k) en la "lattice" ideal [12].

5.4 Algoritmos CGI para Compensación de Movimiento

Sullivan, et. al., [4], propusieron utilizar interpolación bilineal en el campo de desplazamiento, determinando el vector de movimiento para cada pel, al interpolar los valores de los vectores de desplazamiento de sus cuatro puntos de control alrededor. Esto requiere dos interpolaciones bilineales por pel (una para la componente horizontal del vector de movimiento y una para su componente vertical) y, producirá en general, vectores de movimiento no enteros.

Así, como con cualquier método de compensación de movimiento con desplazamientos fraccionales, debemos también interpolar intensidades de pel de la trama previa. Se usará entonces, otra interpolación bilineal para determinar la intensidad predicha.

El valor de un pixel es entonces determinado por tres interpolaciones bilineales, dos para el vector de movimiento y una para la intensidad. Sullivan denomina a este tipo de CGI, CGI tri-bilineal (TBCGI).

La interpolación suave del campo de desplazamiento utilizada en TBCGI preserva la continuidad espacial y la conectividad de la imagen de entrada.

En vez de simplemente trasladar bloques grandes de pixels vecinos, la TBCGI ajusta, estira, traslada y rota secciones cuadriláteras de la imagen de entrada, para formar la imagen compensada de predicción. Consecuentemente, el "zoom" y la rotación se pueden representar ambos, por la TBCGI, sin perder conectividad a lo largo de los bordes de los objetos.

Se pueden probar varios vectores de movimiento candidatos para cada punto de control y elegir el que proporcione la distorsión más baja.

Dado que un vector de movimiento elegido para un nodo de control afectará a todos los pixels en una región delimitada por sus ocho puntos de control adyacentes, se debe implementar un algoritmo que realice esta elección de una manera óptima. Si el tamaño de bloque de la rejilla de control es de $w \times h$ pixels, el vector de movimiento escogido para cada punto de control interno afecta a $(2w-1)x(2h-1)$ valores de pixel.

Debido a la interacción entre los vectores de movimiento, un cambio en un vector de movimiento de un punto de control puede afectar la elección de vectores de movimiento óptimos para sus puntos de control adyacentes.

Esto nos conduce a pensar, que si pudiésemos probar todas la combinaciones de posibles vectores de movimiento, para cada punto de control, encontraríamos una forma segura de encontrar un mínimo global del error de predicción. En la práctica, esto resulta casi imposible de llevarse a cabo.

Sullivan [4], propone una técnica iterativa muy eficiente. En este método, se asocia una bandera llamada "local_opt" con cada punto o nodo de control. Los vectores de movimiento iniciales y las banderas "local_opt" son inicializados todos a cero, indicando con esto que los vectores de movimiento de los puntos de control pueden no ser localmente óptimos.

Los puntos de control son procesados secuencialmente en cada iteración. Si una bandera local_opt de un punto de control es cero, su vector de movimiento es buscado utilizando una búsqueda local, manteniendo constantes los vectores de movimiento para sus puntos de control adyacentes.

Los algoritmos de búsqueda desarrollados para BMA's, se pueden utilizar para desempeñar este tipo de búsqueda.

Si un vector de movimiento cambia de valor, entonces su bandera local_opt y todas las banderas de los puntos de control adyacentes son colocadas en cero. Si por el contrario, el vector de movimiento no cambia, su bandera local_opt asociada es cambiada a un valor de uno. El algoritmo termina cuando todas las banderas local_opt son iguales a uno, o cuando se alcance un número máximo permitido de iteraciones.

En [4], Sullivan dá un seudocódigo de este algoritmo el cual se reproduce a continuación para mayor claridad.

5.4.1 Algoritmo de refinamiento para los vectores de movimiento de los nodos de control

```
Poner todas las banderas local_opt en cero
Poner todos los vectores de movimiento de los nodos de control en cero
Poner iteraciones en cero
Repetir
        Incrementar iteraciones
        Poner count_checks a cero
        For i = 1 to max_i    (cada renglón de puntos de control)
              For j = 1 to max_j    (cada columna de nodos de control)
                    If local_opt(i,j) = = 0
                          Incrementar count_checks
                          Poner MV_old = MV(i,j)
                          Poner MV(i,j) a un valor óptimo local
                          If MV_old = = MV(i,j)
                                Poner local_opt(i,j) = 1
                          Else
                                For m = -1 to 1 and n = -1 to 1
                                      Set local_opt (i + m, j + n) = 0
Until count_checks = = 0.

END
```

NOTA: En este seudocódigo, la abreviatura MV significa vector de movimiento.

En [4], Sullivan, propone un algoritmo en el que todos los vectores de movimiento de los puntos de control, se fijan inicialmente en cero, para después ser refinados iterativamente como ya se dijo antes. Posteriormente, el algoritmo de búsqueda utilizado para la CGI prueba los $(2d + 1)^2$ vectores en la región circundante al valor utilizado en la iteración previa (MV_old) con un desplazamiento máximo de $\pm d$ en las direcciones x y y. El rango de los vectores de movimiento es restringido también a $\pm p$ en cada dirección (En [4] Sullivan prueba con p= 11, y d=3). Este método de búsqueda es similar en espíritu a la búsqueda de *uno a la vez*, excepto que este método busca en una región grande en cada paso, y por esta razón resulta menos susceptible de caer en mínimos locales.

Los métodos TBCGI son siempre superiores en desempeño a los métodos tradicionales de apareamiento de bloques (BMA's), ya que pueden compensar mucho mejor las deformaciones y movimientos complejos (acercamientos, rotaciones, etc.) de los objetos en la escena, y no sólamente los movimientos puramente traslacionales. Por tanto, se pueden obtener menores distorsiones que con los BMA's.

5.5 Desarrollo de un método TBCGI para compensación de movimiento, con condiciones iniciales

5.5.1 Introducción

El método que Sullivan presenta en [4], es muy pesado, computacionalmente hablando, porque debe realizar un proceso de minimización del error de predicción del área delimitada por los ocho nodos que circundan a cada punto de control en la imagen. Esta minimización se efectúa mediante una búsqueda exhaustiva en la ventana de $d \times d$ centrada en la última posición en que quedó el vector en la iteración anterior. Dado que el método de Sullivan comienza con todos los nodos de control en sus posiciones de desplazamiento cero (es decir, con vectores de valor cero), resulta altamente tardado, puesto que debe realizar compensación de movimiento para determinar el error de predicción de cada tentativa de vector, correspondiente a cada una de las posiciones de cada una de las ventanas de búsqueda, de cada uno de los nodos de control, para cada una de las iteraciones del proceso.

Es posible, en la práctica, mejorar notablemente el método de Sullivan si antes de iniciar el algoritmo, se dán, a manera de condiciones iniciales, valores adecuados de vectores de desplazamiento de los nodos de control, de tal forma que el algoritmo se encuentre más cerca de su convergencia desde el principio.

Ahora bien, una primera aproximación al campo de desplazamiento de los nodos de una red, puede ser obtenida si se utiliza un algoritmo BMA convencional en el que cada vector inicial para cada nodo, sea determinado usando un bloque cuadrado cuyo centro represente a dicho vector. Así pues, se dividirá la trama actual en bloques de por ejemplo 8x8 y se realizará un BMA para cada uno de estos bloques. Los vectores de movimiento obtenidos, representarán los valores iniciales de una red cuyos nodos están centrados en los centros geométricos de cada bloque situado en su posición de mínima distorsión sobre la trama pasada.

Posteriormente, estos vectores se utilizarán como valores iniciales de la red (en lugar de cero) para comenzar a ejecutar el algoritmo TBCGI.

5.5.2 Coherencia de una red deformable

Teóricamente, el algoritmo deSullivan busca encontrar un conjunto o una ordenación específica de vectores de nodos de la red cuadrilátera, tal que con dicha ordenación se obtenga un mínimo global en el error de predicción (cuando se compara la

trama compensada con la original). Ahora bien, estas posiciones óptimas se pueden obtener por un método iterativo como el de Sullivan en el que se van buscando mínimos locales para todos lo nodos en cada iteración, hasta que la red en sí deje de presentar cambios de una iteración a la siguiente. Si el proceso se hizo correctamente, dicha red debe representar aproximadamente al movimiento que ocurrió de una trama a la otra, como una deformación ligera de trama, es decir, la red se recorre o se deforma ligeramente para formar una imagen compensada muy parecida a la original. Esto supone que cada nodo está muy relacionado con sus ocho vecinos, entonces su posición no puede ser independiente de la de los demás.

Por tanto, un método BMA podría funcionar bien como alternativa de generador de condiciones iniciales si las tramas en cuestión carecen de ruido, y las texturas son lo suficientemente no homogeneas como para no confundir al BMA y que exista una correlación de los vectores del campo de desplazamiento.

Sin embargo, sabemos, que dado que la tarea de los BMA es encontrar un mínimo error de predicción para este esquema y no obtener un campo de desplazamiento coherente, resulta imposible asegurar coherencia del campo de desplazamiento para tramas reales de video. Lo más probable con un BMA es que el campo de desplazamiento obtenido se encuentre altamente desordenado y por tanto muy diferente al campo real de desplazamiento.

De esta manera, una alternativa más sensata de dar condiciones iniciales al algoritmo TBCGI, sería utilizar un campo de desplazamiento muy parecido al campo de desplazamiento real, en el que obviamente los vectores están altamente correlacionados.

5.5.3 Implementación de un método de campo coherente

Para poder hablar de la coherencia de un campo, necesitamos hablar de la relación de un vector con sus vecinos más cercanos. Si esta relación no existe, es decir, cada valor de vector es independiente de sus vecinos en el campo, entonces el campo no es coherente y está decorrelacionado. Dado que el campo de desplazamiento real en una secuencia debe ser coherente, se implementará un método que dé por resultado un campo coherente.

Una manera de dar coherencia a un campo es "forzar" a un vector determinado a parecerse, o a tener relación con sus ocho vecinos (si pensamos en conectividad ocho).

5.5.4 Método de coherencia de campo por medio de una búsqueda sesgada

El campo de desplazamiento obtenido mediante un método convencional BMA puede utilizarse como base para aplicar algún método de suavizado de campo, o algún método que fuerce la coherencia del campo. El método que se describirá a continuación obliga a cada vector de un campo obtenido con un BMA común, a tener relación con sus ocho vectores vecinos más cercanos. Dicho método tiene por objeto obtener un campo coherente sin perder definición en los bordes de los objetos móviles, es decir, sin permitir que el campo se suavice en las fronteras de movimiento, y sin perder la parte de independencia que podría tener un vector con respecto a sus vecinos (si se trata por ejemplo del movimiento de un objeto pequeño que sólo concierne al vector central).

El primer paso, consiste en suponer que cada posición marcada por cada uno de los ocho vectores vecinos, a partir de la posición central tiene una probabilidad de ocurrencia con una distribución normal asociada.

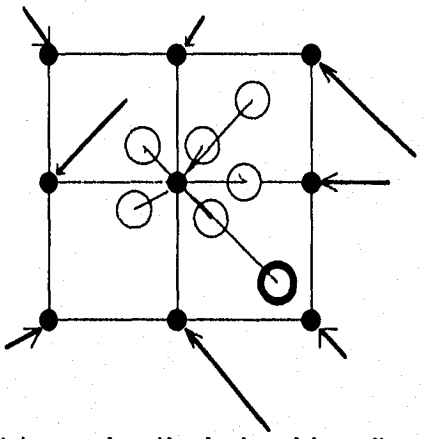


Figura 5.3 Zonas de máxima ponderación donde se debe realizar la búsqueda tipo BMA

En la figura 5.3 se muestra cada vector asociado a cada nodo vecino de la red. Además, cada circulo sin rellenar representa la posición central de una distribución normal relacionada con una posición de alguno de los ocho vectores vecinos.

Primero se realizará un algoritmo BMA convecional para tener un campo inicial de vectores de movimiento. Terminado esto, se procede a realizar otro BMA en el cual se emplea el algoritmo de búsqueda sesgada.

Así pues, el algoritmo de búsqueda sesgada (ABS) consiste en realizar un algoritmo BMA común , pero en el cual, el proceso de búsqueda se pondera o se "carga" hacia ciertas posiciones de la ventana de búsqueda en donde se cree que es más probable encontrar un mínimo local.

Introduciremos brevemente el pricipio matemático en el que se basa este algoritmo. Consideremos una variable aleatoria unidimensional continua **X** la cual puede tomar valores entre $-\infty$ y $+\infty$, luego supongamos que tenemos dos eventos distintos para los cuales conocemos las distribuciones condicionales de X cuando se dá algunos de los dos eventos. Es decir, consideremos el evento A y el evento B, mutuamente excluyentes, y las siguientes probabilidades condicionales,

Probabilidad de que x = a, dado que sólo se dió el evento A = P(x=a| A)

Probabilidad de que x = b, dado que sólo se dió el evento B = P(x=b| B)

Entonces, la probabilidad de que X = x pudiéndose haber dado cualquiera de los dos eventos A o B, es decir la probabilidad total está dada por

$$P(X = x) = pP(x|A) + (1 - p)P(x|B) \tag{5.5}$$

donde p es la probabilidad del evento A, es decir, P(A) = p. Por tanto la probabilidad de B es P(B) = 1-p.

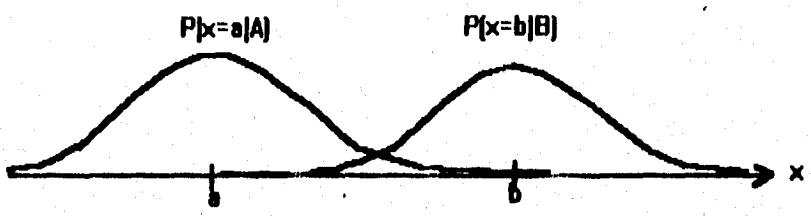


Figura 5.4. . Distribuciones de probabilidad condicionales para cada evento

Generalizando ahora, si se tienen *n* eventos mutuamente excluyentes y todos tienen la misma probabilidad de realizarse, entonces la probabilidad para cada uno será de *1/n.*

Según el teorema de Bayes, la densidad de probabilidad total para la variable X, estará dada por

$$f(x) = \frac{1}{n} f(X = x|A_1) + \ldots + \frac{1}{n} f(X = x|A_n) \tag{5.6}$$

donde $A_1, A_2, \ldots, A_n$ son cada uno de los eventos.

De esta forma, volviendo al problema que nos concierne, podemos aplicar el teorema de Bayes a nuestro problema de vectores, si consideramos a los eventos *A's* que se han mencionado, cada uno como el evento en que el valor del vector central sólo tiene relación con un vecino específico, digamos

A1 = el valor del vector del nodo central sólo tiene relación con el vecino 1
A2 = el valor del vector del nodo central sólo tiene relación con el vecino 2
:
A8 = el valor del vector del nodo central sólo tiene relación con el vecino 8

En esta forma, la probabilidad condicional de que el vector del nodo central se encuentre en una determinada posición puede ser modelada como

$$P(V = \bar{v} | A_j) = \frac{1}{\sigma_j} g\left(\frac{\left| \bar{v} - \bar{\eta}_j \right|}{\sigma_j} \right) \tag{5.7}$$

donde $\bar{v}$ representa la posición con respecto al nodo central, $\bar{\eta}_j$ representa la posición media o central de la distribución que coincide precisamente con el vector del vecino j , g representa una distribución gaussiana estandar [11]. Por último, σ_j es una desviación estandar cuyo valor se asocia al hecho de que el vector del nodo central no necesariamente debe ser igual a su vecino j si se dió el evento Aj. Dando con esto cierta libertad al vector central de ser algo distinto al vecino en cuestión.

Si por ejemplo $\sigma_j = 0$, entonces habiéndose dado el evento *Aj*, el valor del vector central debe ser exactamente igual a su vecino j . Esto supondría una gran correlación entre vectores contiguos. Si por otro lado, $\sigma_j \to \infty$, entonces la elección del valor del vector central es independiente de su vecino j, es decir, dependería únicamente de la búsqueda tipo BMA común.

Por las razones anteriormente expuestas, de aquí en adelante deberemos llamar al valor σ_j *parámetro de semejanza entre vectores vecinos* (PSVV).

También, es de hacerse notar que cuando el PSVV es igual a cero, la decisión se vuelve dura, es decir, se escogerá sólo alguno de los ocho vectores vecinos, dependiendo de en cual de estas ocho posiciones se encuantra la menor distorsión cuando se realiza el BMA (la búsqueda sólo se hará en estas ocho posiciones para este caso).

Si en cambio el PSVV es grande, la decisión es suave y tendrá más posibilidades de ser escogida la media de los ocho vectores.

Por último, si el PSVV es muy grande, la posibilidad o probabilidad de escoger cualquier valor sobre la ventana de búsqueda será la misma y sólo dependerá del lugar en donde la mínima distorsión sea encontrada con el BMA.

El valor de PSVV será único en el algoritmo, y es un dato que se puede ir ajustando hasta encontrar el óptimo que proporciona la mejor PSNR entre la trama compensada y la original.

En la práctica, computacionalmente hablando, se debe crear una matriz de *l x l* tal que *l* sea impar (como 5 o 7) y que contenga los valores de una campana Gaussiana bidimensional.

Se construye una matriz de probabilidades de la misma dimensión que la matriz de distorsiones resultante de la busqueda exhaustiva con un BMA.

Para ello, se van sumando una por una las ocho campanas correspondientes a los ocho vectores vecinos. Cada campana está centrada en la posición de cada vector vecino. Recuérdese que cada vector vecino debe estar referenciado al centro geométrico de la vecindad.

Una vez que la matriz de probabilidades ha sido creada, se calcula otra matriz cuyos elementos son $P'_{i,j} = 1 - P_{i,j}$, , donde $P_{i,j}$ es la probabilidad en la localidad (i,j) de la matriz de probabilidades. Esta nueva matriz se debe multiplicar elemento por elemento con la matriz de distorsiones hallada con el método convencional BMA de búsqueda exhaustiva. Una vez hecho este producto de elementos, se busca el mínimo y su posición será el vector estimado.

En la práctica, al principio del programa se crea una Gaussiana estandar evaluándose la siguiente expresión

$$P(r) = \frac{1}{8} e^{\left(-\frac{r^2}{2(PSVV)^2}\right)} \qquad (5.8)$$

donde *r* es la distancia desde la posición (x,y) al centro de la matriz.

5.5.5 Propuesta de un método alternativo al de Sullivan

Como un método alternativo y más veloz que el algoritmo TBCGI propuesto por Sullivan en [4], en este trabajo de tesis se propone también la implantación de un método TBCGI, el cual emplea un proceso aleatorio de minimización local de error de predicción, para alcanzar iterativamente también un mínimo global de error.

Dicho método utiliza en cada iteración el algoritmo de la búsqueda conjugada propuesto en [6] para efectuar la búsqueda del mínimo error local en cada nodo de la red.

El proceso de intentar minimizar el error global de predicción sobre toda la trama, impone la necesidad de ir variando el valor de cada vector de cada nodo de control sobre una ventana de búsqueda, y para cada valor de prueba de dicho vector, formar una imagen compensada (usando TBCGI) de la región que delimitan los ocho vecinos del vector del nodo central.

Si la búsqueda es exhaustiva en dicha ventana de búsqueda, el tiempo que consuma el algoritmo, dependerá del tamaño de aquella. Por esta razón, en [4] se proponen ventanas pequeñas (d=3), que de todas formas son eficientes dada la característica iterativa del algoritmo.

Sin embargo, si se utiliza una búsqueda conjugada (en [6]) partiendo de la última posición (obtenida en la iteración pasada) de un vector, es plausible ahorrarse un gran número de búsquedas, debido también, al hecho de que en las próximas iteraciones el valor del vector a buscar será cada vez más cercano al valor del vector anterior si suponemos que el algoritmo debe tender a la convergencia.

Se debe recordar que cuando se busca el vector óptimo de un nodo de la red, se considera que los ocho vectores de sus ocho nodos vecinos, están fijos , tal como si ellos ya hubieran sido optimizados.

Sin embargo, si se optimiza el valor de un vector de nodo y luego se procede a optimizar el siguiente (es decir, su vecino) la optimización del primero queda desajustada puesto que el valor del vecino con el que se hizo dicha otimización, ha cambiado.

De esta forma, cuando ajustamos un lado estamos inconcientemente desajustando ligeramente otro lado.

Experimentalmente, se observó que el error de predicción global no decrece cuando en una iteración se realiza la minimización en el mismo orden que en la siguiente iteración. Es decir, si el orden en que se atiende a cada nodo es el mismo que en la iteración siguiente, es muy posible que el error global no tienda a decrecer.

En este tipo de esquemas de optimización global, se ha visto que los errorres tienden a sumarse en forma geométrica, de manera tal que si se sigue siempre el mismo orden en las minimizaciones locales por nodo, los errorres se acumulan más y más, debido a que como siempre se sigue el mismo patrón de orden, los errores no se pueden anular unos a otros. Como son de la misma clase siempre, y clases iguales se suman, el error global no tiene por que disminuir globalmente.

Por la razón anterior, se propuso un método de minimización global aleatorio. Este método consiste en seguir un orden aleatorio distinto en cada iteración para realizar las búsquedas de error mínimo por nodo de control en la rejilla.

Además, con el fin de agilizar el algoritmo de optimización, se utilizó el método de la búsqueda conjugada para realizar las búsquedas en cada nodo.

Capítulo 6

Pruebas y Resultados

6.1 Introducción

Este capítulo, presenta todas las pruebas realizadas, tanto para algoritmos de apareamiento de bloques (BMA's), como para algoritmos TBCGI. Se verá con claridad, la utilidad de los primeros como complemento de los segundos (en la compensación por interpolación TBCGI), para la obtención de vectores de movimiento que funjen como condiciones iniciales, y que son muy próximos a los vectores de los nodos de una malla ideal cuyo error de predicción global es mínimo.

6.2 Secuencias de video utilizadas en las pruebas.

Las secuencias de video utilizadas para todas las pruebas realizadas en este trabajo de tesis fueron:

1. La secuencia "Interview" que consta de tramas de 536 renglones por 674 columnas.
2. La secuencia "Cal_train" que consta de tramas de220 renglones por 280 columnas
3. La secuencia "Sintética" que consta de tramas de 256 renglones por 256 columnas.

Estas son secuencias con una gran cantidad de movimiento y están originalmente sin formato (imágenes tipo "raw"). Cada pixel es de 256 niveles de gris, en 8 bits.

De estas secuencias, se utilizaron sólo regiones de 256x256, en las que aparecen rasgos importantes de las mismas. En la secuencia "segmentada" *interview* aparece una joven sentada en un sillón de sala que se levanta hasta quedar casi completamente de pie. En la secuencia Cal_train, aparecen varios objetos tales como un tren eléctrico de juguete que se desplaza horizontalmente hacia la izquierda, mientras una pelota que un principio fue empujada por éste rueda por delante. Al mismo tiempo, aparece un calendario de pared paralelo al plano de la imagen que se desplaza diagonalmente hacia arriba.

Finálmente, en la secuencia sintética, aparecen tres figuras planas con movimiento paralelo al plano de la imagen, sobre un fondo tipo tablero de ajedrez inmóvil. Tales figuras son: un rectángulo grande que se mueve a la derecha, un círculo grande que crece a manera de "zoom", un rectángulo alargado y pequeño de tono claro que se desplaza hacia la izquierda cubriendo parte del círculo, y finálmente un cuadrado oscuro y pequeño en la parte inferior derecha, que presenta un movimiento de rotación.

En seguida, se muestra un par de tramas de cada una de las secuencias utilizadas.

Figura 6.1 Tramas 5 y 6 de la secuencia "interview"

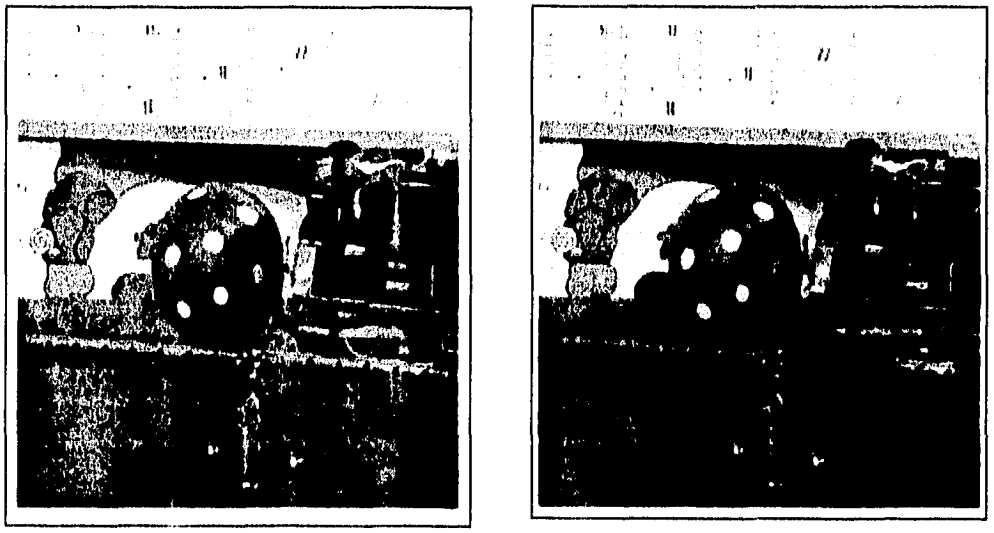


Figura 6.2 Tramas 34 y 35 de la secuencia "Cal_train"

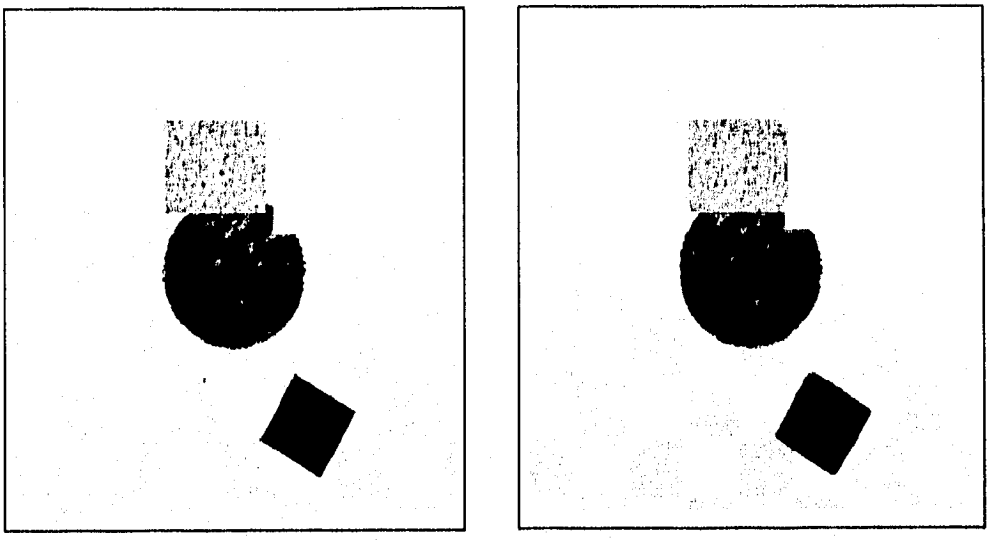


Figura 6.3 Tramas 13 y 14 de la secuencia "sintética"

6.3 Programación de los algoritmos de compensación de movimiento por apareamiento de bloques

La programación de los algoritmos de apareamiento de bloques fue realizada en lenguaje Turbo C 3.1 de Microsoft. Y todas las ejecuciones de estos algoritmos se efectuaron sobre una plataforma PC de IBM con microprocesador 80486 de 40 Mhz de frecuencia de reloj.

Se programó un algoritmo de transmisión para cada uno de los siguientes métodos:

1. Búsqueda exhaustiva
2. Búsqueda logarítmica bidimensional.
3. Búsqueda en la dirección conjugada.
4. Búsqueda OTS modificada.

El algoritmo de transmisión mencionado es un programa que tiene como entrada los siguientes datos:

1. Las tramas pasada y presente
2. El tamaño de los bloques (MxM, donde M es una potencia de 2, como 4,8,16)
3. El máximo desplazamiento entre tramas $d_{máx}$ (de este valor depende el tamaño de la ventana de búsqueda)

4. Las dimensiones de las tramas (256x256 para todos los casos)

La salida del programa de transmisión está compuesta de:

1. Un archivo que contiene los vectores de movimiento, con un byte para cada componente de un vector

El programa de recepción es común para cualquiera de los algoritmos mencionados de transmisión. Este programa crea una imagen de predicción (o compensada) de la trama presente. Los datos de entrada son:

1. Un archivo que contenga los vectores de movimiento
2. La dimensión de las tramas (256x256)
3. El tamaño del bloque (8, por ejemplo)
4. La trama pasada.

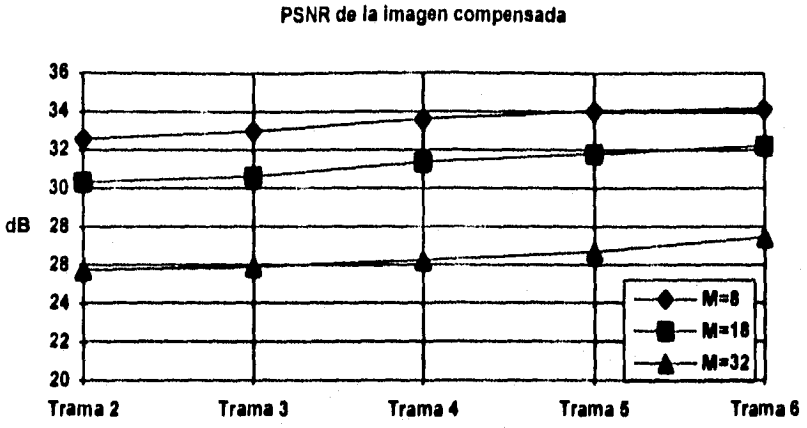
La salida del programa de recepción es una imagen compensada.

También se implantó un programa que calcula el error cuadrático medio y la relación señal a ruido pico PSNR en decibeles, entre la trama de predicción y la trama presente original. Además este programa crea una imagen de error para fines de visualización.

6.4 Resultados obtenidos con los BMA's.

Se hicieron algunas gráficas que muestran la relación señal a ruido pico de las imágenes de predicción obtenidas con los programas anteriormente mencionados, para 6 tramas de la secuencia "interview".

La gráfica 6.1 muestra el desempeño del método de la búsqueda exhustiva para distintos tamaños de bloque, y $d_{máx} = 7$. En esta gráfica, se hace patente que cuando se aumenta el tamaño del bloque, el error crece debido a que se vuelve menos válida la hipótesis de que todos los pixels dentro de un bloque siguen el mismo desplazamiento.

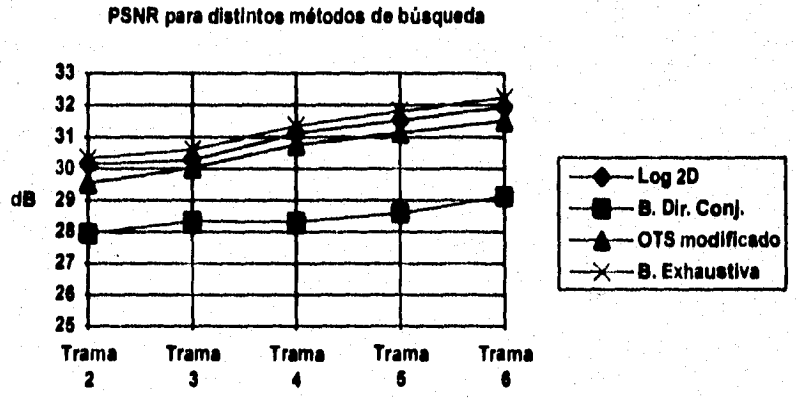


Gráfica 6.1 PSNR de las tramas de predicción para 3 tamaños de bloque

Como un siguiente paso en esta serie de pruebas, se tiene la evaluación del desempeño de los distintos tipos de búsqueda detallados en el capítulo 4.

Los algoritmos evaluados fueron el algoritmo de búsqueda logarítmica 2D, el método de búsqueda en la dirección conjugada, y el método OTS modificado.

En la gráfica 6.2 se muestra la PSNR de las imágenes de predicción obtenidas, con un tamaño de bloque de 16x16.



Gráfica 6.2 Desempeño de los distintos tipos de búsqueda para bloques de 16x16

En esta última gráfica se puede apreciar que tanto el algoritmo de la búsqueda logarítmica 2-D, como el OTS modificado proporcionan un buen desempeño en cuanto a PSNR se refiere. Sin embargo, en cuanto a rapidez, el algoritmo de la búsqueda conjugada es el mejor.

La siguiente tabla muestra un estudio comparativo de los tiempos relativos de ejecución de los cuatro métodos de búsqueda. Los tiempos fueron tomados para un tamaño de bloque de 8x8, y usando las tramas 5 y 6 de la secuencia "interview".

MÉTODO	TIEMPO RELATIVO (La B. Exhaustiva tomó 100 segundos)
Búsqueda Exhaustiva	1
Búsqueda Logarítmica 2D	0.188
Búsqueda en la Dirección Conjugada	0.167
Búsqueda OTS modificada	0.161

Tabla 6.1 Tiempos relativos de procesamiento de los algoritmos de búsqueda

6.4.1 Problemas de "bloque" de los BMA's

Una característica poco agradable de los BMA's es su tendencia a presentar artefactos de bloque. Esto se debe como ya se ha mencionado antes, a la discontinuidad que se crea en el campo de desplazamiento cuando los vectores de dos bloques contiguos son muy distintos. Este defecto de los BMA's se hace más evidente cuando no se emplea el método de búsqueda exhaustiva. La figura siguiente muestra en (a) la trama de predicción de la trama 6 de la secuencia "interview", obtenida con el método de la búsqueda en la dirección conjugada, y usando bloques de 16x16. En (b) se puede observar la señal de error en la que se aprecian los artefactos de bloque inherentes a los BMA's.

(a) (b)

Figura 6.4 (a) Trama Compensada, (b) Imagen de error

6.5 Programación de los algoritmos de compensación de movimiento por TBCGI.

6.5.1 Importancia de la coherencia en los campos de desplazamiento

El diseño de un algoritmo de compensación de movimiento debe cuidar no sólo la relación señal a ruido que posee y la calidad visual subjetiva de las imágenes compensadas, sino también que la entropía del campo de vectores de desplazamiento sea lo más pequeña posible, ya que los vectores de movimiento deben ser enviados al receptor y lo más conveniente es que sean codificados con algún algoritmo compresor sin pérdidas, tal como un codificador de Huffman, el cual es un codificador entrópico.

Dado que la entropía representa la cota mínima de compresión posible cuando se codifica por Huffman, es de gran importancia cuidar que la entropía por vector de movimiento, dada en [bits/vector], sea mínima.

Un campo vectorial desordenado como el que se genera comunmente con los métodos BMA, puede tener una entropía mucho mayor que aquel generado con algún método que cuide la semejenza de los vectores vecinos, tal como el método de la búsqueda sesgada presentado en el capítulo 5.

Además de los puntos mencionados, se debe cuidar otro aspecto fundamental en la implementación de los algoritmos de compensación de movimiento: Este aspecto es el tiempo. La rapidez de ejecución en la simulación de un algoritmo se verá reflejada en la implantación de éste en "hadrware" de tiempo real.

Por tanto, se deben cuidar cuatro aspectos básicos : la PSNR, la calidad visual subjetiva, la entropía de campo de desplazamiento, y la rapidez de ejecución.

6.5.2 TBCGI sin optimización del campo de desplazamiento

En [4], se propuso un método de compensación que consta de un transmisor y un receptor.

El transmisor realiza la siguiente función:

Coloca nodos o puntos de control igualmente espaciados que forman una red o rejilla cuadrilátera sobre la trama presente. A cada nodo se le asigna un vector cero al

principio. Luego, mediante una serie de iteraciones, va ajustando los vectores de los nodos hasta que el error de predicción TBCGI alcance un mínimo global.

El receptor, recibe los vectores de movimiento de los nodos, generados por el transmisor y forma una imagen compensada usando TBCGI.

El tiempo de procesamiento del transmisor propuesto en [4] es extremadamente grande, debido a que cada vez que se prueba un vector de un nodo, se debe formar una imagen compensada por TBCGI, de la región que delimitan los ocho nodos vecinos al nodo en cuestión. De este modo, si en una iteración se realizan, por ejemplo 9 búsquedas para cada nodo (usando ventanas de búsqueda de 3x3), se deberán realizar 9 compensaciones con TBCGI para cada nodo en una iteración. Si por ejemplo, la imagen es de 256x256 y se colocan nodos espaciados a 8 pixels unos de otros, tendremos aproximadamente 1024 nodos sobre toda la imagen. Se efectuarán entonces 9216 compensaciones en una sólo iteración. Generalmente se requeire más de una iteración para alcanzar una buena PSNR, así que el número de compensaciónes seguramente ascenderá a más del doble de esta cifra.

Por las razones anteriormente expuestas, en esta tesis se realizó el siguiente experimento: Tomar como nodos de control los centros geométricos de los bloques que utilizan los BMA's ya programados. Luego, utilizar los vectores generados por un BMA de búsqueda exhaustiva para darlos como datos de entrada a un receptor, el cual construye una imagen compensada con TBCGI, y que considera que los nodos de control están colocados justo en la forma ya mencionada.

Se usó la secuencia "Cal_train" para este experimento, por ser la que presenta una mayor cantidad de objetos pequeños naturales. Los bloque para el BMA se ajustaron a un tamaño de 8x8, y el desplazamiento $d_{máx}$ se hizo igual a 10. Para la transmisión se usó un BMA de búsqueda exhaustiva llamado BMA_EXHA.C, y para la recepción uno llamado RX_TBCGI.C.

El resultado de este experimento fue una trama compensada con una PSNR = 31.00 dB y una ENTROPIA = 5.55 bits/vector.

Un experimento usando de nuevo BMA_EXHA.C para la transmisión, y para la recepción un algoritmo de compensación BMA (llamado RX_BMA), produce una PSNR = 31.92 dB, y una entropia de 5.55. Este representa el caso convencional *transmisión-recepción* tipo BMA.

Se puede ver de los resultados anteriores, que los vectores obtenidos con un BMA pueden ser usados como una aproximación algo burda de los que se obtendrían utilizando optimización TBCGI. Sin embargo se obtienen los siguientes beneficios:

1. Dado que la compensación se hace en un receptor TBCGI, se eliminan los artefactos de bloque (porque el campo de desplazamiento es continuo)

2. El tiempo de ejecución se reduce, ya que no se usa optimización TBCGI en el transmisor como en [4].
3. La PSNR es bastante alta, aunque no alcanza el nivel que se obtiene usando un compensador BMA.

Utilización del método de coherencia de campo (búsqueda sesgada)

En el experimento anterior, se utilizó como aproximación de un campo optimizado, un campo obtenido con un BMA convencional, y se vió que aunque la entropía es alta, se obtienen beneficios en tiempo y en eliminación de artefactos de bloque.

Es posible mejorar los resultados anteriores, si damos como entrada al compensador TBCGI (receptor) un campo de desplazamiento coherente, que en teoría podría ser más parecido al que produciría un algoritmo de optimización TBCGI como el mencionado en [4].

Se realizó el experimento siguiente: primero se utilizó el programa de transmisión BMA, llamado BMA_EXHA.C, y luego se efectuó la búsqueda sesgada (programa B_SESG.C) sobre el campo de vectores generado por el programa anterior. Finalmente, una vez que el campo es coherente, se realiza la compensación TBCGI usando el programa RX_TBCGI.C.

Un $PSVV^2 = 3.5$ en la búsqueda sesgada proporcionó el mejor resultado en la PSNR. Los resultados son los siguintes:

PSNR = 31.6 dB
Eliminación de Artefectos de bloque
ENTROPIA = 3.3 bits/vector

Por tanto, se puede pensar que un campo coherente puede considerarse como una mejor aproximación a la que se obtendría por optimización TBCGI. Además proporciona un nivel excelente de entropía del campo de desplazamiento, debido a que el campo está áltamente ordenado.

La figura 6.5 ilustra las mejoras obtenidas con este método sobre el esquema *transmisión - recepción* BMA convencional.

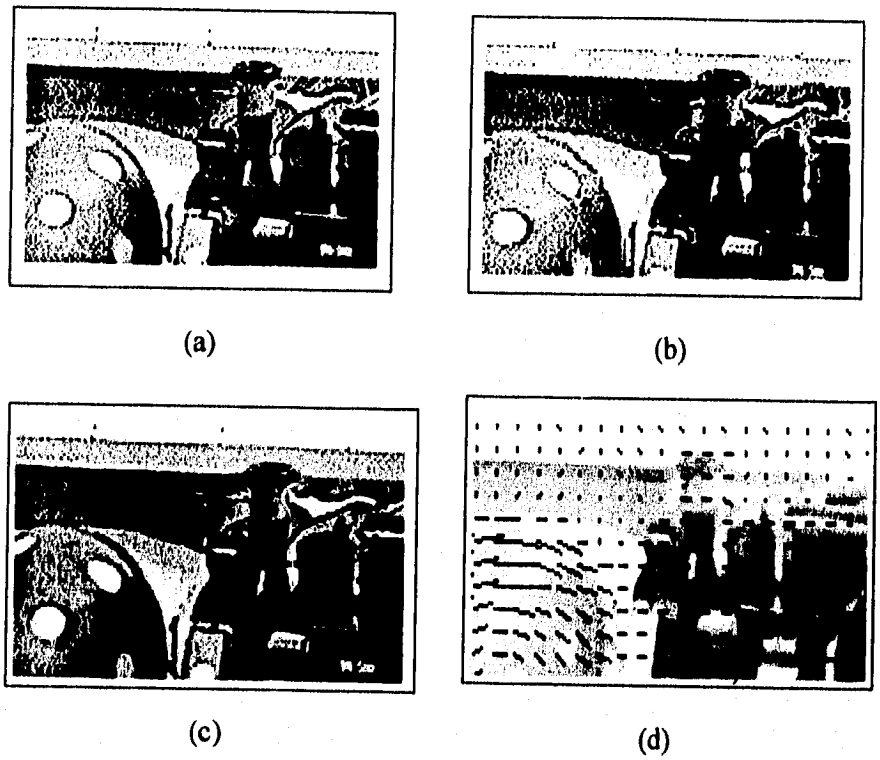


Figura 6.5 (a) Acercamiento de la trama 35 original (b) Predicción hecha con esquema convencional BMA. (c) Predicción hecha con TBCGI, usando los vectores de la búsqueda sesgada BMA. (d) Campo de desplazamiento obtenido con búsqueda sesgada.

En la figura 6.5b se puede apreciar el "efecto de bloque" del esquema BMA convencional. En (c) se aprecia una supresión total del efecto de bloque al utilizar un sistema de compensación TBCGI. Finalmente en (d) se ilustra el campo coherente que sirvió para realizar la compensación TBCGI. La entropía de este campo es ideal para el uso de un codificador de Huffman.

En la figura 6.6b se muestra el campo de movimiento obtenido con el programa de búsqueda sesgada para las tramas 36 y 37 de la secuencia "Cal_train".

Se puede apreciar que el campo es altamente coherente sin suavizar las fronteras de movimiento como sucedería si se usára un filtro de media sobre el campo.

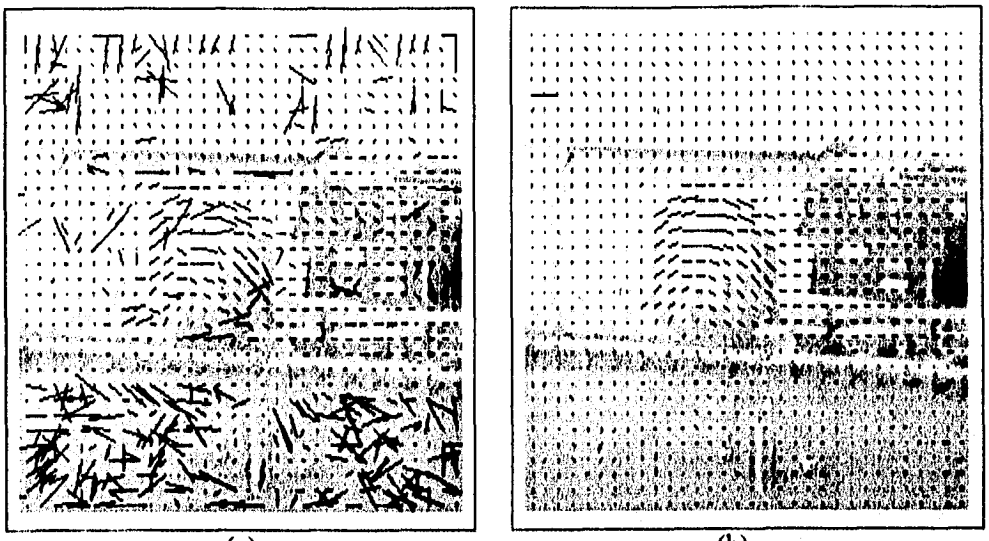


Figura 6.6. (a) Campo vectorial obtenido con un BMA de búsqueda exhaustiva. (b) Campo obtenido con el método de búsqueda sesgada.

6.5.3 Compensación TBCGI con optimización del campo de desplazamiento

El esquema de compensación de la sección anterior, antepone la ventaja de la baja entropía y la rapidez de ejecución, a la desventaja de una PSNR no muy alta.

Ahora bien, si lo que se desea es conseguir una PSNR alta, sin importar el tiempo de ejecución, lo que se tiene que hacer es utilizar un transmisor que genere los vectores para los nodos de la rejilla de control, por medio de un algoritmo de optimización TBCGI, ya que el receptor es TBCGI también.

Para este fin, se programaron dos algoritmos de transmisión que realizan esta tarea: 1. El algoritmo de Sullivan propuesto en [4] programado en "lenguaje C" con nombre SULLIVAN.C y 2. El algoritmo TBCGI de optimización aleatoria propuesto en esta tesis, cuyo programa hecho en "lenguje C" lleva el nombre de CGI_ALEA.C.

En los experimentos que a continuación se mencionan, se usaron 4 tramas consecutivas de la secuencia "Cal_train" (tramas 34,35,36,37), y 3 tramas consecutivas de la secuencia sintética (tramas 10,11,12).

Las pruebas se dividieron en 4 tipos : A, B, C, D.

Prueba A

La prueba tipo A, consistió en utilizar el algoritmo TBCGI de optimización aleatoria de 4 iteraciones. A este algoritmo se le dán como entrada los vectores generados por el programa BMA_EXHA. Posteriormente, los vectores ya optimizados deben darse como entrada al programa receptor RX_TBCGI.C, que genera la imagen compensada.

Prueba B

En esta prueba se usa también el algoritmo TBCGI de Optimización Aleatoria de 4 iteraciones. A este algoritmo se le dán como entrada los vectores generados por el programa de búsqueda sesgada B_SESG.C, el cual a su vez recibió los vectores del programa BMA_EXHA.C. La salida del algoritmo TBCGI de Optimización Aleatoria, se debe dar como entrada al programa receptor RX_TBCGI.C, para que genere la trama de predicción.

Prueba C

Esta prueba consistió en utilizar el algoritmo de optimización TBCGI de Sullivan tal cual, que genera un campo optimizado de salida, el cual es dado como entrada al programa de recepción RX_TBCGI.C, que crea la compensación.

Prueba D

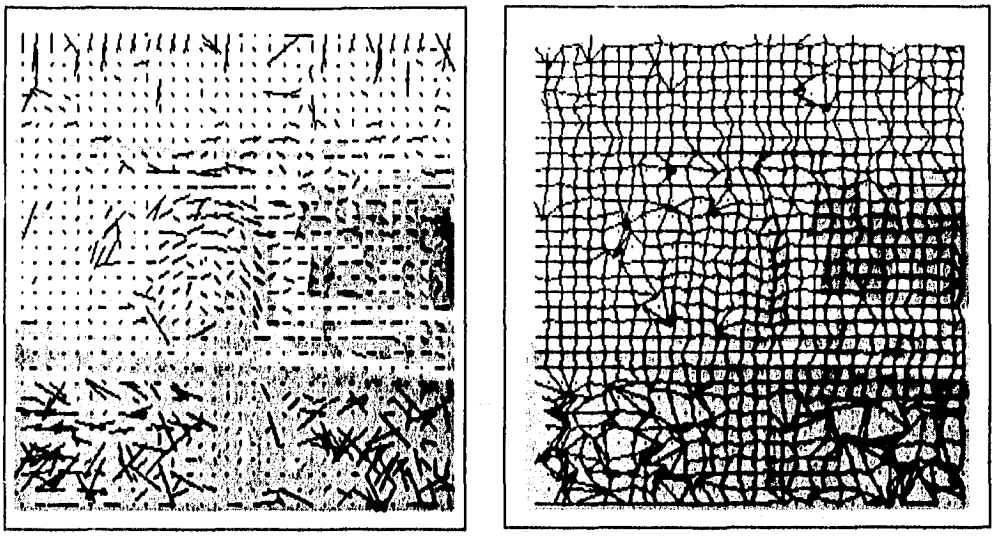
Por último, se utilizan los vectores que genera el programa de búsqueda sesgada B_SESG.C (que a su vez recibió como entrada los generados por BMA_EXHA.C), para darlos como entrada (a manera de condiciones iniciales) al algoritmo de Sullivan. Después, con los vectores generados por el algoritmo de Sullivan, se hará la compensación de movimiento a través de programa RX_TBCGI.C

Secuencia "Cal_train"

Prueba A

TRAMA	PSNR (dB)	Tiempo (minutos)	ENTROPIA (bits/vector)	No. de Iteraciones
35	33.3	22:00	5.61	4
36	33.10	22:40	5.79	4
37	33.00	22:40	5.74	4

Tabla 6.2 Resultados de la prueba A



(a) (b)

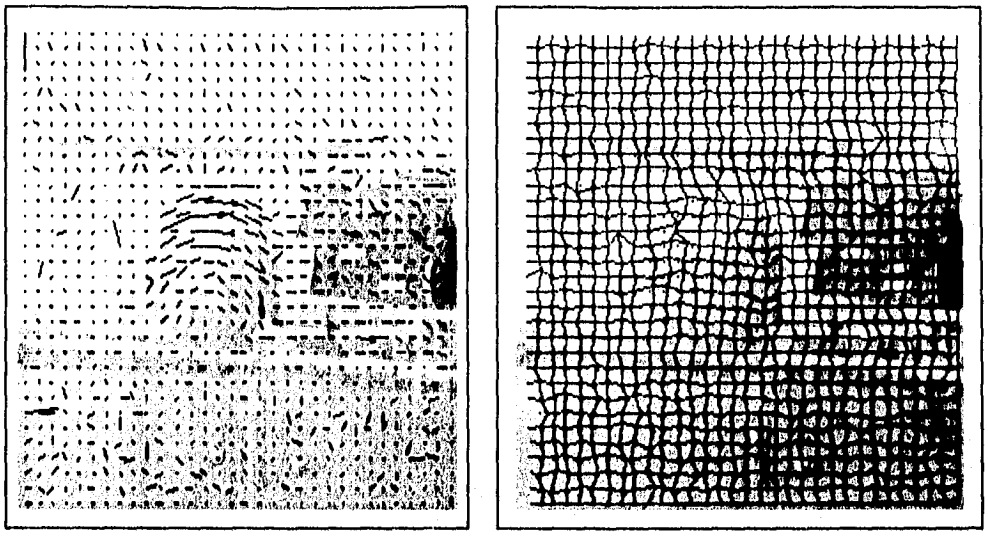
Figura 6.7 Resultados de la prueba A. (a) Campo de desplazamiento final (b) Rejilla de mapeo de la trama pasada

Los resultados de la tabla 6.2 muestran que aunque la PSNR es alta, el campo de desplazamiento no refleja el movimiento real de los objetos en la escena. Además la entropía es bastante alta.

Prueba B

TRAMA	PSNR (dB)	Tiempo (minutos)	ENTROPIA (bits/ vector)	No. de Iteraciones
35	33.74	22:00	4.4	4
36	33.63	22:40	4.2	4
37	33.84	22:45	4.35	4

Tabla 6.3 Resultados de la prueba B



(a) (b)

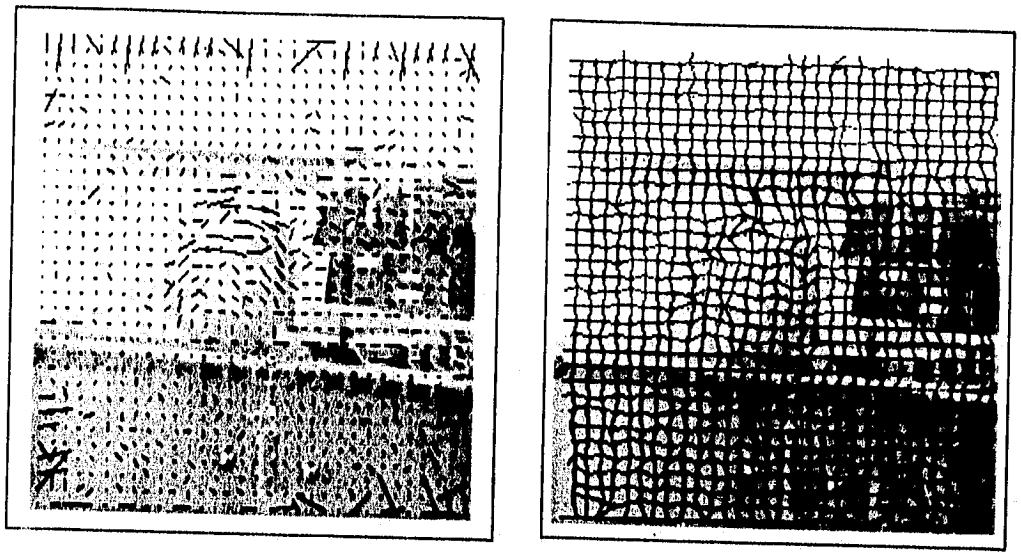
Figura 6.8 Resultados de la prueba B. (a) Campo de desplazamiento final (b) Rejilla de mapeo de la trama pasada

Los resultados de la tabla 6.3 revelan una mejora notable en cuanto a la PSNR, con el beneficio adicional de tener ahora una entropía mucho menor que en el experimento pasado. El hecho de haber dado como entrada un campo altamente coherente al programa de optimización aleatoria TBCGI, genera un campo que aunque no está tan correlacionado, refleja más el movimiento real de los abjetos y está mejor relacionado con el principio de deformación de rejilla del método TBCGI. Recuérdese que en este método, una red o rejilla irregular (trama pasada) se "mapea" a una rejilla regular (trama presente compensada).

Prueba C

TRAMA	PSNR (dB)	Tiempo (minutos)	ENTROPIA (bits/ vector)	No. de Iteraciones
35	33.75	48:00	4.67	13
36	33.65	46:00	4.64	13
37	33.75	48:00	4.66	19

Tabla 6.4 Resultados de la prueba C



(a) (b)

Figura 6.9 Resultados de la prueba C. (a) Campo de desplazamiento final (b) Rejilla de mapeo de la trama pasada

El algoritmo de Sullivan proporciona una buena relación señal a ruido y una baja entropía. Sin embargo, el hecho de no dar vectores iniciales, ocasiona que el algoritmo tarde mucho tiempo en converger.

Prueba D

TRAMA	PSNR (dB)	Tiempo (minutos)	ENTROPIA (bits/ vector)	No. de Iteraciones
35	33.72	38:00	4.37	11
36	33.72	35:00	4.36	9
37	33.97	39:00	4.39	10

Tabla 6.5 Resultados de la prueba D

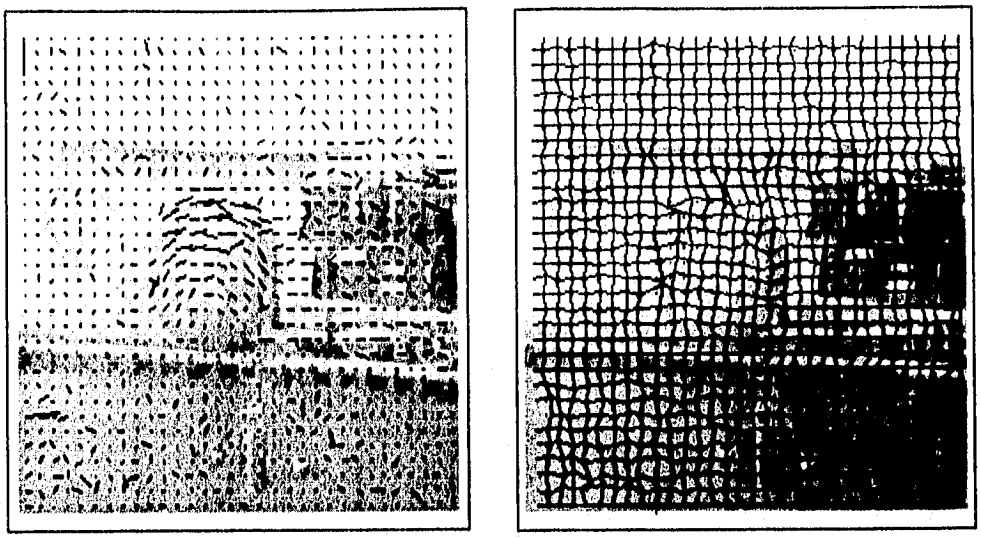


Figura 6.10 Resultados de la prueba D. (a) Campo de desplazamiento final (b) Rejilla de mapeo de la trama pasada

De los resultados en la tabla 6.5, se puede ver que una buena mejora al método de Sullivan consiste en dar como condiciones iniciales, los vectores del campo de desplazamiento coherente generado por una búsqueda sesgada. El tiempo de procesamiento es menor y la entropía también. Esto se debe a que al haber dado como condición inicial para comenzar la optimización, un campo de muy baja entropía, tomando en cuenta que el mínimo global será encontrado mucho antes, los vectores no tendrán tiempo de "desacomodarse" lo suficiente como para aumentar su entropía.

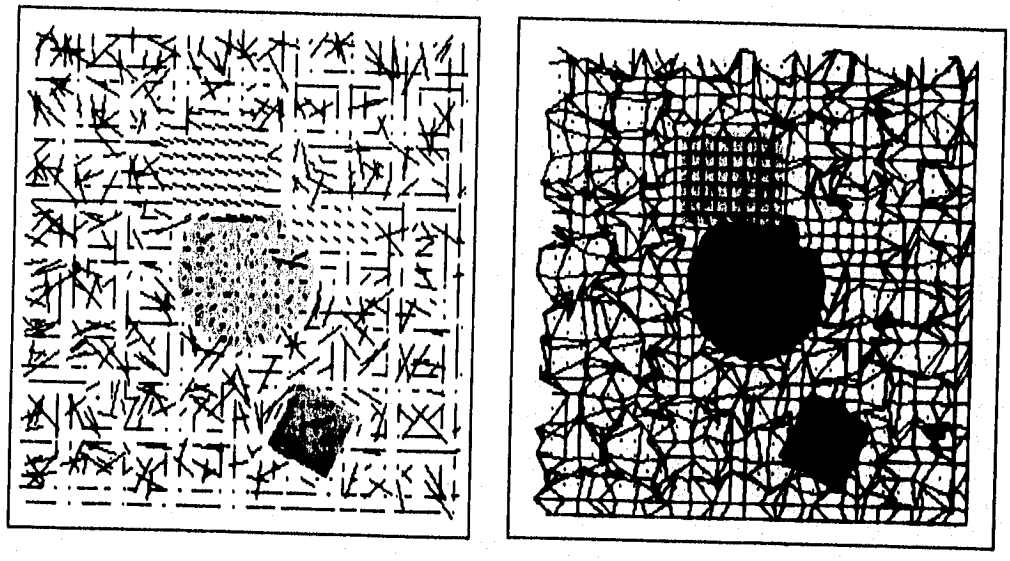
De las cuatro pruebas (A, B, C, D), se puede concluir que la prueba B es la que ofrece mejores resultados en general, ya que el tiempo de ejecución es el más pequeño de todas las pruebas, la entropía es muy baja y la PSNR es tan alta como en el algoritmo de Sullivan.

Secuencia "Sintética"

Prueba A

TRAMA	PSNR (dB)	Tiempo (minutos)	ENTROPIA (bits/ vector)	No. de Iteraciones
11	36.3	24:00	6.87	4
12	35.62	24:17	6.97	4

Tabla 6.6 Resultados de la prueba A



(a) (b)

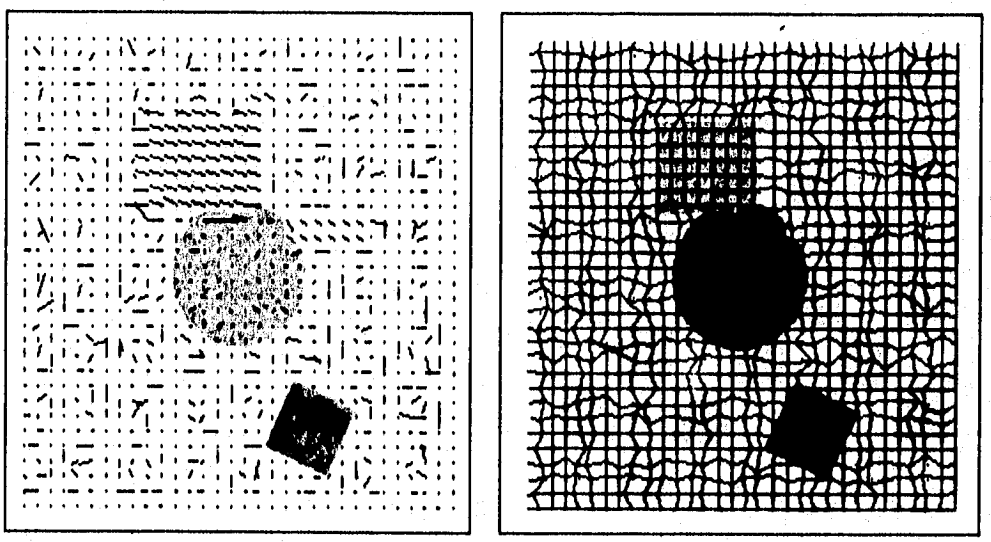
Figura 6.11 Resultados de la prueba A. (a) Campo de desplazamiento final (b) Rejilla de mapeo de la trama pasada

Al igual que en la secuencia "Cal_train", los resultados de la tabla 6.6 y de la figura 6.11, muestran que aunque la PSNR es alta, el campo de desplazamiento no refleja el movimiento real de los objetos en la escena. Además la entropía es muy alta.

Prueba B

TRAMA	PSNR (dB)	Tiempo (minutos)	ENTROPIA (bits/vector)	No. de Iteraciones
11	37.45	23:00	4.1	4
12	36.86	22:42	4.22	4

Tabla 6.7 Resultados de la prueba B



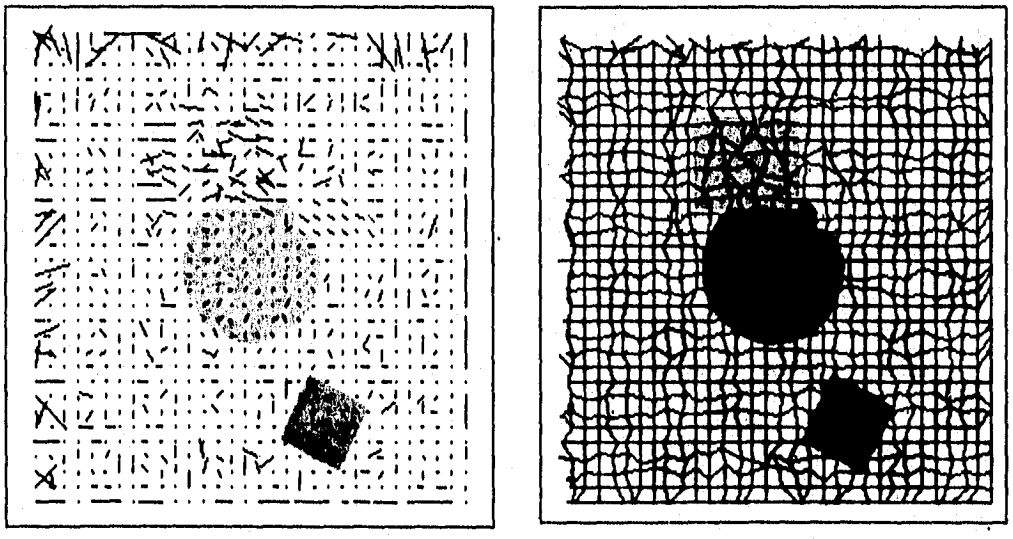
(a) (b)

Figura 6.12 Resultados de la prueba B. (a) Campo de desplazamiento final (b) Rejilla de mapeo de la trama pasada

Prueba C

TRAMA	PSNR (dB)	Tiempo (minutos)	ENTROPIA (bits/ vector)	No. de Iteraciones
11	35.4	49:35	4.72	12
12	34.75	47:00	4.88	14

Tabla 6.8 Resultados de la prueba C



(a) (b)

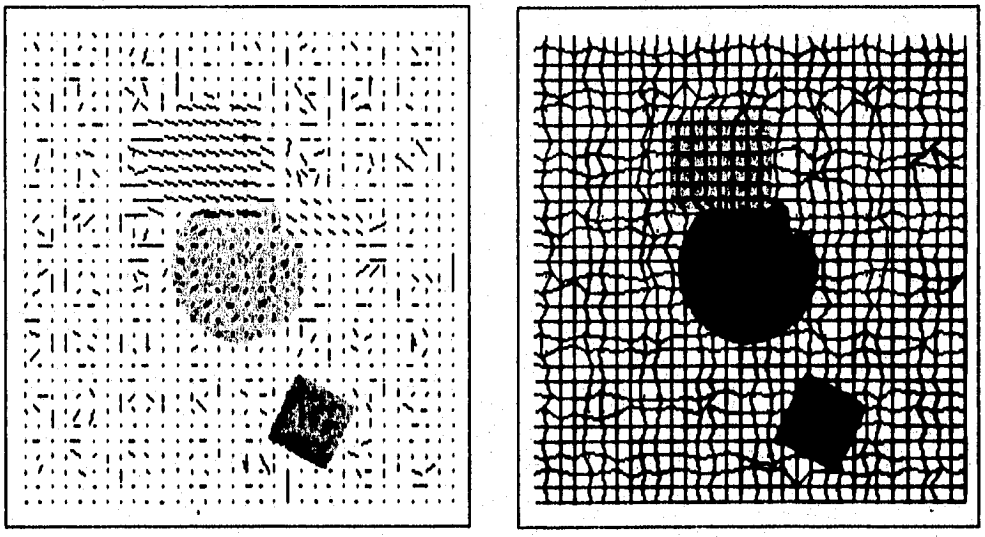
Figura 6.13 Resultados de la prueba C. (a) Campo de desplazamiento final (b) Rejilla de mapeo de la trama pasada

Igualmente volvemos a constatar que, el algoritmo de Sullivan proporciona una buena relación señal a ruido y una baja entropía. Sin embargo, el hecho de no dar vectores iniciales ocasiona que el algoritmo tarde mucho tiempo en converger.

Prueba D

TRAMA	PSNR (dB)	Tiempo (minutos)	ENTROPIA (bits/vector)	No. de Iteraciones
11	37.51	40:00	4.05	12
12	36.93	40:00	4.18	15

Tabla 6.9 Resultados de la prueba D



(a) (b)

Figura 6.14 Resultados de la prueba D. (a) Campo de desplazamiento final (b) Rejilla de mapeo de la trama pasada

Capítulo 7

Conclusiones

El uso de la interpolación bilineal por rejillas de control en la compensación de movimiento, proporciona una significativa mejora sobre los métodos tradicionales de compensación de movimiento por apareamiento de bloques, en virtud de la obtención de un campo de desplazamiento suave y coherente, el cual no presenta discontinuidades como en el caso de los primeros métodos de apareamiento. El tener un campo suave implica que los efectos de bloque característicos en los BMA's, serán anulados completamente en los métodos TBCGI. Por otra parte, el número de vectores de movimiento en uno y otro métodos es exactamente el mismo lo que supone una compresión igual para ambos.

Además, la falta de coherencia encontrada en un campo de desplazamiento obtenido en un BMA implica una entropía alta que a fin de cuentas no ayudaría a obtener una buena tasa de compresión. En el método propuesto TBCGI, se busca lograr una entropía pequeña para este campo vectorial. Esto se logra, al utilizar un método de coherencia de campo como el que se propone en esta tesis.

Este campo coherente de muy baja entropía, al ser usado como condición inicial en un método de optimización TBCGI de error de predicción, dá como resultado un gran ahorro de tiempo de procesamiento, lo cual no se logra cuando las condiciones iniciales son cero, como en el método de Sullivan.

Aquí se propone también un algoritmo de optimización alternativo al de Sullivan, el cual alcanza prácticamente los mismos niveles de PSNR en tan sólo 4 iteraciones, y por ende en un tiempo de procesamiento mucho menor.

Dado que el algoritmo no realiza demasiadas iteraciones, la baja entropía propia del campo de desplazamiento inicial (condición inicial) no se incrementa demasiado cuando el algoritmo ha terminado de procesar. De hecho, esta entropía, según se pudo apreciar en los resultados, es menor que la que se obtiene con el método de Sullivan, en el que las condiciones iniciales son cero.

También se debe decir, que dado que a final de cuentas, este algoritmo implica una transformación geométrica de cuadriláteros deformes en la trama anterior a cuadrilateros regulares en la trama presente, el hecho de tener un campo de desplazamiento coherente determina que también la malla o rejilla de deformación será coherente y se ajustará mucho mejor al movimiento tridimensional de los objetos de la escena. Los movimientos de traslación, los acercamientos, los alejamientos, e incluso la rotación tridimensional en cualquier eje, se modelarán mejor que en los métodos BMA's.

Finalmente, se debe agregar que la implantación de este tipo de métodos por rejillas deformables, implica siempre una ganancia en la compensación de movimiento debido a que la malla se ajusta o se deforma de acuerdo al movimiento de los objetos. El movimiento tridimensional de los objetos sólidos en la escena, se puede modelar bastante bien, como deformaciones geométricas planas de los cuadriláteros que se forman. (con los nodos de control que residen dentro de dichos objetos).

A manera de una semblanza de lo que se podría hacer como continuación de este trabajo, se puede mencionar que una mejora en general a estos métodos se alcanzaría si se decidiera colocar los nodos de control en posiciones estratégicas en la imagen (como podrían ser los bordes de los objetos), y no en forma regular sobre la trama presente como se suele hacer en los métodos que a esta tesis conciernen. La compensación de movimiento sería mucho mejor, puesto que cada cuadrilátero pertenecería a un solo objeto en la escena y no se compartirían dos objetos en un sólo cuadrilátero como aquí sucede.

A pesar de que este esquema con seguimiento de bordes superaría (en cuanto a PSNR se refiere) al esquema tratado en este trabajo, el tiempo de procesamiento para el primero sería mucho más elevado que para el segundo.

De una u otra forma, queda aún mucho camino por recorrer para llegar al diseño de algoritmos de compensacióin de movimiento que además de modelarlo fielmente, sean capaces de hacerlo eficazmente en el tiempo más corto posible y así poder implantarlos en los procesadores modernos, para su ejecución en tiempo real.

Bibliografía

[1] H.Li, A. Lundmark, y R. Forchheimer, "Image Sequence Coding at Very Low Bitrates: A review," en *IEEE transactions on image processing*, vol. 3, no. 5, septiembre 1994.

[2] H. G. Musmann, P. Pirsch, y H. J. Grallert, "Advances en Picture Coding," en *Proc. IEEE*, vol. 73, no. 4, pp. 523-548, Abril 1985.

[3] A. K. Jain, "Image Data Compression: A Review," en *Proc. IEEE*, vol. 69, no. 3, pp. 349-389, Marzo 1981.

[4] G. J. Sullivan y Richard L. Baker, "Motion Compensation for Video Compression using Control Grid Interpolation," en *Proc. IEEE ICASSP 91*, 1991.

[5] Q. Wang y R. J. Clarke, "Motion Estimation and Compensation for Image Sequence Coding," en *Signal Processing: Image Communications 4*, pp. 161-174, Elsevier, 1992.

[6] R. Srinivasan y K. R. Rao, "Predictive Coding Based on Efficient Motion Estimation," en *IEEE Transactions on communications*, vol. com-33, no. 8, Agosto 1985.

[7] R. Srinivasan y K. R. Rao, "Motion-Compensated Coder for Videoconferencing," en *IEEE transactions on communications*, vol. com-35, no. 3, Marzo de 1987.

[8] J. R. Jain y A. K. Jain, "Displacement Measurement and Its Application in Interframe Image Coding," en *IEEE transactions on communications*, vol. com-29, no. 12, Diciembre 1981.

[9] A. N. Netravali y B. G. Haskell, "Digital Pictures, Representation and Compression." Plenum Press, New York 1988.

[10] A. Murat Tekalp, "Digital Video Processing." Prentice Hall PTR, NJ 1995.

[11] A. Papoulis, "Probability, Random Variables, and Stochastic Processes." McGraw-Hill International Editions, 1991.

[12] Lira, Jorge, "Procesamiento Digital de Imágenes y Reconocimiento de Patrones." Instituto de Geofísica, UNAM, México D.F.